

activemq-cpp-3.1.2

Generated by Doxygen 1.6.2-20100208

Fri Apr 30 14:04:20 2010

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Data Structure Index	3
2.1	Class Hierarchy	3
3	Data Structure Index	21
3.1	Data Structures	21
4	File Index	51
4.1	File List	51
5	Namespace Documentation	69
5.1	activemq Namespace Reference	69
5.1.1	Detailed Description	69
5.2	activemq::cmsutil Namespace Reference	70
5.3	activemq::commands Namespace Reference	71
5.4	activemq::core Namespace Reference	73
5.5	activemq::exceptions Namespace Reference	74
5.6	activemq::io Namespace Reference	75
5.7	activemq::library Namespace Reference	76
5.8	activemq::state Namespace Reference	77
5.9	activemq::threads Namespace Reference	78
5.10	activemq::transport Namespace Reference	79
5.11	activemq::transport::correlator Namespace Reference	80
5.12	activemq::transport::failover Namespace Reference	81
5.13	activemq::transport::inactivity Namespace Reference	82
5.14	activemq::transport::logging Namespace Reference	83
5.15	activemq::transport::mock Namespace Reference	84
5.16	activemq::transport::tcp Namespace Reference	85

5.17	activemq::util Namespace Reference	86
5.18	activemq::wireformat Namespace Reference	87
5.19	activemq::wireformat::openwire Namespace Reference	88
5.20	activemq::wireformat::openwire::marshal Namespace Reference	89
5.21	activemq::wireformat::openwire::marshal::v1 Namespace Reference	90
5.22	activemq::wireformat::openwire::marshal::v2 Namespace Reference	94
5.23	activemq::wireformat::openwire::marshal::v3 Namespace Reference	98
5.24	activemq::wireformat::openwire::marshal::v4 Namespace Reference	102
5.25	activemq::wireformat::openwire::marshal::v5 Namespace Reference	106
5.26	activemq::wireformat::openwire::utils Namespace Reference	110
5.27	activemq::wireformat::stomp Namespace Reference	111
5.28	cms Namespace Reference	112
5.28.1	Detailed Description	114
5.29	decaf Namespace Reference	115
5.29.1	Detailed Description	115
5.30	decaf::internal Namespace Reference	116
5.31	decaf::internal::io Namespace Reference	117
5.32	decaf::internal::net Namespace Reference	118
5.33	decaf::internal::nio Namespace Reference	119
5.34	decaf::internal::util Namespace Reference	120
5.35	decaf::internal::util::concurrent Namespace Reference	121
5.36	decaf::io Namespace Reference	122
5.37	decaf::lang Namespace Reference	124
5.37.1	Function Documentation	125
5.37.1.1	operator!=	125
5.37.1.2	operator!=	125
5.37.1.3	operator==	125
5.37.1.4	operator==	125
5.38	decaf::lang::exceptions Namespace Reference	126
5.39	decaf::net Namespace Reference	127
5.40	decaf::nio Namespace Reference	128
5.41	decaf::security Namespace Reference	129
5.42	decaf::security::auth Namespace Reference	130
5.43	decaf::security::auth::x500 Namespace Reference	131
5.44	decaf::security::cert Namespace Reference	132
5.45	decaf::security_provider Namespace Reference	133

5.46	decaf::security_provider::unix Namespace Reference	134
5.47	decaf::security_provider::unix::openssl Namespace Reference	135
5.48	decaf::util Namespace Reference	136
5.49	decaf::util::comparators Namespace Reference	138
5.50	decaf::util::concurrent Namespace Reference	139
5.51	decaf::util::concurrent::atomic Namespace Reference	141
5.52	decaf::util::concurrent::locks Namespace Reference	142
5.53	decaf::util::logging Namespace Reference	143
5.53.1	Enumeration Type Documentation	143
5.53.1.1	Level	143
5.54	std Namespace Reference	145
6	Data Structure Documentation	147
6.1	decaf::util::AbstractCollection< E > Class Template Reference	147
6.1.1	Detailed Description	149
6.1.2	Constructor & Destructor Documentation	150
6.1.2.1	~AbstractCollection	150
6.1.3	Member Function Documentation	150
6.1.3.1	add	150
6.1.3.2	addAll	151
6.1.3.3	clear	151
6.1.3.4	contains	152
6.1.3.5	containsAll	152
6.1.3.6	copy	153
6.1.3.7	equals	153
6.1.3.8	isEmpty	153
6.1.3.9	lock	154
6.1.3.10	notify	154
6.1.3.11	notifyAll	154
6.1.3.12	operator=	154
6.1.3.13	remove	155
6.1.3.14	removeAll	155
6.1.3.15	retainAll	156
6.1.3.16	toArray	157
6.1.3.17	tryLock	157
6.1.3.18	unlock	157
6.1.3.19	wait	158

6.1.3.20	wait	158
6.1.3.21	wait	159
6.1.4	Field Documentation	159
6.1.4.1	mutex	159
6.2	decaf::util::AbstractList< E > Class Template Reference	160
6.2.1	Detailed Description	160
6.2.2	Constructor & Destructor Documentation	160
6.2.2.1	~AbstractList	160
6.3	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference	161
6.3.1	Detailed Description	161
6.3.2	Constructor & Destructor Documentation	161
6.3.2.1	~AbstractMap	161
6.4	decaf::util::AbstractQueue< E > Class Template Reference	162
6.4.1	Detailed Description	162
6.4.2	Constructor & Destructor Documentation	163
6.4.2.1	AbstractQueue	163
6.4.2.2	~AbstractQueue	163
6.4.3	Member Function Documentation	163
6.4.3.1	add	163
6.4.3.2	addAll	163
6.4.3.3	clear	164
6.4.3.4	element	164
6.4.3.5	remove	164
6.5	decaf::util::AbstractSequentialList< E > Class Template Reference	166
6.5.1	Detailed Description	166
6.5.2	Constructor & Destructor Documentation	166
6.5.2.1	~AbstractSequentialList	166
6.6	decaf::util::AbstractSet< E > Class Template Reference	167
6.6.1	Detailed Description	167
6.6.2	Constructor & Destructor Documentation	167
6.6.2.1	~AbstractSet	167
6.6.3	Member Function Documentation	167
6.6.3.1	removeAll	167
6.7	activemq::transport::AbstractTransportFactory Class Reference	169
6.7.1	Detailed Description	169
6.7.2	Constructor & Destructor Documentation	169

6.7.2.1	~AbstractTransportFactory	169
6.7.3	Member Function Documentation	169
6.7.3.1	createWireFormat	169
6.8	activemq::core::ActiveMQAckHandler Class Reference	171
6.8.1	Detailed Description	171
6.8.2	Constructor & Destructor Documentation	171
6.8.2.1	~ActiveMQAckHandler	171
6.8.3	Member Function Documentation	171
6.8.3.1	acknowledgeMessage	171
6.9	activemq::commands::ActiveMQBlobMessage Class Reference	172
6.9.1	Constructor & Destructor Documentation	173
6.9.1.1	ActiveMQBlobMessage	173
6.9.1.2	~ActiveMQBlobMessage	173
6.9.2	Member Function Documentation	173
6.9.2.1	clone	173
6.9.2.2	cloneDataStructure	173
6.9.2.3	copyDataStructure	173
6.9.2.4	equals	174
6.9.2.5	getDataStructureType	174
6.9.2.6	getMimeType	174
6.9.2.7	getName	174
6.9.2.8	getRemoteBlobUrl	175
6.9.2.9	isDeletedByBroker	175
6.9.2.10	setDeletedByBroker	175
6.9.2.11	setMimeType	175
6.9.2.12	setName	175
6.9.2.13	setRemoteBlobUrl	175
6.9.2.14	toString	176
6.9.3	Field Documentation	176
6.9.3.1	BINARY_MIME_TYPE	176
6.9.3.2	ID_ACTIVEMQBLOBMESSAGE	176
6.10	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference	177
6.10.1	Detailed Description	177
6.10.2	Constructor & Destructor Documentation	178
6.10.2.1	ActiveMQBlobMessageMarshaller	178
6.10.2.2	~ActiveMQBlobMessageMarshaller	178

6.10.3	Member Function Documentation	178
6.10.3.1	createObject	178
6.10.3.2	getDataStructureType	178
6.10.3.3	looseMarshal	178
6.10.3.4	looseUnmarshal	179
6.10.3.5	tightMarshal1	179
6.10.3.6	tightMarshal2	179
6.10.3.7	tightUnmarshal	180
6.11	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	
	Class Reference	181
6.11.1	Detailed Description	181
6.11.2	Constructor & Destructor Documentation	182
6.11.2.1	ActiveMQBlobMessageMarshaller	182
6.11.2.2	~ActiveMQBlobMessageMarshaller	182
6.11.3	Member Function Documentation	182
6.11.3.1	createObject	182
6.11.3.2	getDataStructureType	182
6.11.3.3	looseMarshal	182
6.11.3.4	looseUnmarshal	183
6.11.3.5	tightMarshal1	183
6.11.3.6	tightMarshal2	183
6.11.3.7	tightUnmarshal	184
6.12	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	
	Class Reference	185
6.12.1	Detailed Description	185
6.12.2	Constructor & Destructor Documentation	186
6.12.2.1	ActiveMQBlobMessageMarshaller	186
6.12.2.2	~ActiveMQBlobMessageMarshaller	186
6.12.3	Member Function Documentation	186
6.12.3.1	createObject	186
6.12.3.2	getDataStructureType	186
6.12.3.3	looseMarshal	186
6.12.3.4	looseUnmarshal	187
6.12.3.5	tightMarshal1	187
6.12.3.6	tightMarshal2	187
6.12.3.7	tightUnmarshal	188

6.13	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	
	Class Reference	189
6.13.1	Detailed Description	189
6.13.2	Constructor & Destructor Documentation	190
	6.13.2.1 ActiveMQBlobMessageMarshaller	190
	6.13.2.2 ~ActiveMQBlobMessageMarshaller	190
6.13.3	Member Function Documentation	190
	6.13.3.1 createObject	190
	6.13.3.2 getDataStructureType	190
	6.13.3.3 looseMarshal	190
	6.13.3.4 looseUnmarshal	191
	6.13.3.5 tightMarshal1	191
	6.13.3.6 tightMarshal2	191
	6.13.3.7 tightUnmarshal	192
6.14	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	
	Class Reference	193
6.14.1	Detailed Description	193
6.14.2	Constructor & Destructor Documentation	194
	6.14.2.1 ActiveMQBlobMessageMarshaller	194
	6.14.2.2 ~ActiveMQBlobMessageMarshaller	194
6.14.3	Member Function Documentation	194
	6.14.3.1 createObject	194
	6.14.3.2 getDataStructureType	194
	6.14.3.3 looseMarshal	194
	6.14.3.4 looseUnmarshal	195
	6.14.3.5 tightMarshal1	195
	6.14.3.6 tightMarshal2	195
	6.14.3.7 tightUnmarshal	196
6.15	activemq::commands::ActiveMQBytesMessage Class Reference	197
6.15.1	Constructor & Destructor Documentation	200
	6.15.1.1 ActiveMQBytesMessage	200
	6.15.1.2 ~ActiveMQBytesMessage	200
6.15.2	Member Function Documentation	200
	6.15.2.1 clearBody	200
	6.15.2.2 clone	200
	6.15.2.3 cloneDataStructure	200
	6.15.2.4 copyDataStructure	201

6.15.2.5	<code>equals</code>	201
6.15.2.6	<code>getBodyBytes</code>	201
6.15.2.7	<code>getBodyLength</code>	202
6.15.2.8	<code>getDataStructureType</code>	202
6.15.2.9	<code>onSend</code>	202
6.15.2.10	<code>readBoolean</code>	202
6.15.2.11	<code>readByte</code>	203
6.15.2.12	<code>readBytes</code>	203
6.15.2.13	<code>readBytes</code>	204
6.15.2.14	<code>readChar</code>	204
6.15.2.15	<code>readDouble</code>	204
6.15.2.16	<code>readFloat</code>	205
6.15.2.17	<code>readInt</code>	205
6.15.2.18	<code>readLong</code>	205
6.15.2.19	<code>readShort</code>	206
6.15.2.20	<code>readString</code>	206
6.15.2.21	<code>readUnsignedShort</code>	207
6.15.2.22	<code>readUTF</code>	207
6.15.2.23	<code>reset</code>	207
6.15.2.24	<code>setBodyBytes</code>	208
6.15.2.25	<code>toString</code>	208
6.15.2.26	<code>writeBoolean</code>	208
6.15.2.27	<code>writeByte</code>	208
6.15.2.28	<code>writeBytes</code>	209
6.15.2.29	<code>writeBytes</code>	209
6.15.2.30	<code>writeChar</code>	209
6.15.2.31	<code>writeDouble</code>	210
6.15.2.32	<code>writeFloat</code>	210
6.15.2.33	<code>writeInt</code>	210
6.15.2.34	<code>writeLong</code>	211
6.15.2.35	<code>writeShort</code>	211
6.15.2.36	<code>writeString</code>	211
6.15.2.37	<code>writeUnsignedShort</code>	212
6.15.2.38	<code>writeUTF</code>	212
6.15.3	Field Documentation	212
6.15.3.1	<code>ID_ACTIVEMQBYTESMESSAGE</code>	212

6.16	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	
	Class Reference	213
6.16.1	Detailed Description	213
6.16.2	Constructor & Destructor Documentation	214
	6.16.2.1 ActiveMQBytesMessageMarshaller	214
	6.16.2.2 ~ActiveMQBytesMessageMarshaller	214
6.16.3	Member Function Documentation	214
	6.16.3.1 createObject	214
	6.16.3.2 getDataStructureType	214
	6.16.3.3 looseMarshal	214
	6.16.3.4 looseUnmarshal	215
	6.16.3.5 tightMarshal1	215
	6.16.3.6 tightMarshal2	215
	6.16.3.7 tightUnmarshal	216
6.17	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	
	Class Reference	217
6.17.1	Detailed Description	217
6.17.2	Constructor & Destructor Documentation	218
	6.17.2.1 ActiveMQBytesMessageMarshaller	218
	6.17.2.2 ~ActiveMQBytesMessageMarshaller	218
6.17.3	Member Function Documentation	218
	6.17.3.1 createObject	218
	6.17.3.2 getDataStructureType	218
	6.17.3.3 looseMarshal	218
	6.17.3.4 looseUnmarshal	219
	6.17.3.5 tightMarshal1	219
	6.17.3.6 tightMarshal2	219
	6.17.3.7 tightUnmarshal	220
6.18	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	
	Class Reference	221
6.18.1	Detailed Description	221
6.18.2	Constructor & Destructor Documentation	222
	6.18.2.1 ActiveMQBytesMessageMarshaller	222
	6.18.2.2 ~ActiveMQBytesMessageMarshaller	222
6.18.3	Member Function Documentation	222
	6.18.3.1 createObject	222
	6.18.3.2 getDataStructureType	222

6.18.3.3	looseMarshal	222
6.18.3.4	looseUnmarshal	223
6.18.3.5	tightMarshal1	223
6.18.3.6	tightMarshal2	223
6.18.3.7	tightUnmarshal	224
6.19	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	
	Class Reference	225
6.19.1	Detailed Description	225
6.19.2	Constructor & Destructor Documentation	226
	6.19.2.1 ActiveMQBytesMessageMarshaller	226
	6.19.2.2 ~ActiveMQBytesMessageMarshaller	226
6.19.3	Member Function Documentation	226
	6.19.3.1 createObject	226
	6.19.3.2 getDataStructureType	226
	6.19.3.3 looseMarshal	226
	6.19.3.4 looseUnmarshal	227
	6.19.3.5 tightMarshal1	227
	6.19.3.6 tightMarshal2	227
	6.19.3.7 tightUnmarshal	228
6.20	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	
	Class Reference	229
6.20.1	Detailed Description	229
6.20.2	Constructor & Destructor Documentation	230
	6.20.2.1 ActiveMQBytesMessageMarshaller	230
	6.20.2.2 ~ActiveMQBytesMessageMarshaller	230
6.20.3	Member Function Documentation	230
	6.20.3.1 createObject	230
	6.20.3.2 getDataStructureType	230
	6.20.3.3 looseMarshal	230
	6.20.3.4 looseUnmarshal	231
	6.20.3.5 tightMarshal1	231
	6.20.3.6 tightMarshal2	231
	6.20.3.7 tightUnmarshal	232
6.21	activemq::core::ActiveMQConnection Class Reference	233
6.21.1	Detailed Description	235
6.21.2	Constructor & Destructor Documentation	235
	6.21.2.1 ActiveMQConnection	235

6.21.2.2	~ActiveMQConnection	236
6.21.3	Member Function Documentation	236
6.21.3.1	addDispatcher	236
6.21.3.2	addProducer	236
6.21.3.3	addTransportListener	236
6.21.3.4	close	236
6.21.3.5	createSession	237
6.21.3.6	createSession	237
6.21.3.7	destroyDestination	237
6.21.3.8	destroyDestination	238
6.21.3.9	fire	238
6.21.3.10	getClientID	238
6.21.3.11	getConnectionId	238
6.21.3.12	getConnectionInfo	239
6.21.3.13	getExceptionListener	239
6.21.3.14	getMetaData	239
6.21.3.15	isClosed	239
6.21.3.16	isStarted	240
6.21.3.17	onCommand	240
6.21.3.18	oneway	240
6.21.3.19	onException	240
6.21.3.20	removeDispatcher	240
6.21.3.21	removeProducer	241
6.21.3.22	removeSession	241
6.21.3.23	removeTransportListener	241
6.21.3.24	sendPullRequest	241
6.21.3.25	setExceptionListener	241
6.21.3.26	start	242
6.21.3.27	stop	242
6.21.3.28	syncRequest	242
6.21.3.29	transportInterrupted	242
6.21.3.30	transportResumed	242
6.22	activemq::core::ActiveMQConnectionFactory Class Reference	244
6.22.1	Constructor & Destructor Documentation	245
6.22.1.1	ActiveMQConnectionFactory	245
6.22.1.2	ActiveMQConnectionFactory	245

6.22.1.3	<code>~ActiveMQConnectionFactory</code>	245
6.22.2	Member Function Documentation	245
6.22.2.1	<code>createConnection</code>	245
6.22.2.2	<code>createConnection</code>	246
6.22.2.3	<code>createConnection</code>	246
6.22.2.4	<code>createConnection</code>	246
6.22.2.5	<code>getBrokerURL</code>	247
6.22.2.6	<code>getPassword</code>	247
6.22.2.7	<code>getUsername</code>	247
6.22.2.8	<code>setBrokerURL</code>	247
6.22.2.9	<code>setPassword</code>	248
6.22.2.10	<code>setUsername</code>	248
6.23	<code>activemq::core::ActiveMQConnectionMetaData</code> Class Reference	249
6.23.1	Detailed Description	249
6.23.2	Constructor & Destructor Documentation	250
6.23.2.1	<code>ActiveMQConnectionMetaData</code>	250
6.23.2.2	<code>~ActiveMQConnectionMetaData</code>	250
6.23.3	Member Function Documentation	250
6.23.3.1	<code>getCMSMajorVersion</code>	250
6.23.3.2	<code>getCMSMinorVersion</code>	250
6.23.3.3	<code>getCMSProviderName</code>	250
6.23.3.4	<code>getCMSVersion</code>	251
6.23.3.5	<code>getCMSXPropertyNames</code>	251
6.23.3.6	<code>getProviderMajorVersion</code>	251
6.23.3.7	<code>getProviderMinorVersion</code>	252
6.23.3.8	<code>getProviderVersion</code>	252
6.24	<code>activemq::core::ActiveMQConnectionSupport</code> Class Reference	253
6.24.1	Constructor & Destructor Documentation	254
6.24.1.1	<code>ActiveMQConnectionSupport</code>	254
6.24.1.2	<code>~ActiveMQConnectionSupport</code>	255
6.24.2	Member Function Documentation	255
6.24.2.1	<code>getClientId</code>	255
6.24.2.2	<code>getCloseTimeout</code>	255
6.24.2.3	<code>getNextLocalTransactionId</code>	255
6.24.2.4	<code>getNextSessionId</code>	255
6.24.2.5	<code>getNextTempDestinationId</code>	255

6.24.2.6	getPassword	256
6.24.2.7	getProducerWindowSize	256
6.24.2.8	getProperties	256
6.24.2.9	getSendTimeout	256
6.24.2.10	getTransport	256
6.24.2.11	getUsername	257
6.24.2.12	isAlwaysSyncSend	257
6.24.2.13	isUseAsyncSend	257
6.24.2.14	setAlwaysSyncSend	257
6.24.2.15	setClientId	257
6.24.2.16	setCloseTimeout	257
6.24.2.17	setPassword	258
6.24.2.18	setProducerWindowSize	258
6.24.2.19	setSendTimeout	258
6.24.2.20	setUseAsyncSend	258
6.24.2.21	setUsername	258
6.24.2.22	shutdownTransport	259
6.24.2.23	startupTransport	259
6.25	activemq::core::ActiveMQConstants Class Reference	260
6.25.1	Detailed Description	261
6.25.2	Member Enumeration Documentation	261
6.25.2.1	AckType	261
6.25.2.2	DestinationActions	261
6.25.2.3	DestinationOption	261
6.25.2.4	TransactionState	262
6.25.2.5	URIParam	262
6.25.3	Member Function Documentation	262
6.25.3.1	toDestinationOption	262
6.25.3.2	toString	262
6.25.3.3	toString	262
6.25.3.4	toURIOption	262
6.26	activemq::core::ActiveMQConsumer Class Reference	263
6.26.1	Constructor & Destructor Documentation	265
6.26.1.1	ActiveMQConsumer	265
6.26.1.2	~ActiveMQConsumer	265
6.26.2	Member Function Documentation	265

6.26.2.1	acknowledge	265
6.26.2.2	acknowledge	265
6.26.2.3	afterMessageIsConsumed	266
6.26.2.4	beforeMessageIsConsumed	266
6.26.2.5	clearMessagesInProgress	266
6.26.2.6	close	266
6.26.2.7	commit	266
6.26.2.8	deliverAcks	266
6.26.2.9	dequeue	267
6.26.2.10	dispatch	267
6.26.2.11	doClose	267
6.26.2.12	getConsumerId	267
6.26.2.13	getConsumerInfo	268
6.26.2.14	getLastDeliveredSequenceId	268
6.26.2.15	getMessageListener	268
6.26.2.16	getMessageSelector	268
6.26.2.17	isClosed	268
6.26.2.18	isSynchronizationRegistered	269
6.26.2.19	iterate	269
6.26.2.20	receive	269
6.26.2.21	receive	269
6.26.2.22	receiveNoWait	269
6.26.2.23	rollback	270
6.26.2.24	setLastDeliveredSequenceId	270
6.26.2.25	setMessageListener	270
6.26.2.26	setSynchronizationRegistered	270
6.26.2.27	start	270
6.26.2.28	stop	271
6.27	activemq::library::ActiveMQCPP Class Reference	272
6.27.1	Constructor & Destructor Documentation	272
6.27.1.1	ActiveMQCPP	272
6.27.1.2	ActiveMQCPP	272
6.27.1.3	~ActiveMQCPP	272
6.27.2	Member Function Documentation	272
6.27.2.1	initializeLibrary	272
6.27.2.2	initializeLibrary	273

6.27.2.3	operator=	273
6.27.2.4	shutdownLibrary	273
6.28	activemq::commands::ActiveMQDestination Class Reference	274
6.28.1	Constructor & Destructor Documentation	276
6.28.1.1	ActiveMQDestination	276
6.28.1.2	ActiveMQDestination	276
6.28.1.3	~ActiveMQDestination	276
6.28.2	Member Function Documentation	276
6.28.2.1	cloneDataStructure	276
6.28.2.2	copyDataStructure	277
6.28.2.3	createDestination	277
6.28.2.4	createTemporaryName	277
6.28.2.5	equals	278
6.28.2.6	getClientId	278
6.28.2.7	getCMSDestination	278
6.28.2.8	getDataStructureType	278
6.28.2.9	getDestinationType	279
6.28.2.10	getOptions	279
6.28.2.11	getOrderedTarget	279
6.28.2.12	getPhysicalName	279
6.28.2.13	getPhysicalName	279
6.28.2.14	isAdvisory	280
6.28.2.15	isComposite	280
6.28.2.16	isConnectionAdvisory	280
6.28.2.17	isConsumerAdvisory	280
6.28.2.18	isExclusive	280
6.28.2.19	isOrdered	280
6.28.2.20	isProducerAdvisory	281
6.28.2.21	isQueue	281
6.28.2.22	isTemporary	281
6.28.2.23	isTopic	281
6.28.2.24	isWildcard	281
6.28.2.25	setAdvisory	281
6.28.2.26	setExclusive	282
6.28.2.27	setOrdered	282
6.28.2.28	setOrderedTarget	282

6.28.2.29	setPhysicalName	282
6.28.2.30	toString	282
6.28.3	Field Documentation	283
6.28.3.1	advisory	283
6.28.3.2	ADVISORY_PREFIX	283
6.28.3.3	COMPOSITE_SEPARATOR	283
6.28.3.4	CONNECTION_ADVISORY_PREFIX	283
6.28.3.5	CONSUMER_ADVISORY_PREFIX	283
6.28.3.6	DEFAULT_ORDERED_TARGET	283
6.28.3.7	exclusive	284
6.28.3.8	ID_ACTIVEMQDESTINATION	284
6.28.3.9	options	284
6.28.3.10	ordered	284
6.28.3.11	orderedTarget	284
6.28.3.12	physicalName	284
6.28.3.13	PRODUCER_ADVISORY_PREFIX	284
6.28.3.14	QUEUE_QUALIFIED_PREFIX	284
6.28.3.15	TEMP_POSTFIX	284
6.28.3.16	TEMP_PREFIX	284
6.28.3.17	TEMP_QUEUE_QUALIFIED_PREFIX	284
6.28.3.18	TEMP_TOPIC_QUALIFIED_PREFIX	284
6.28.3.19	TOPIC_QUALIFIED_PREFIX	284
6.29	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	
	Class Reference	285
6.29.1	Detailed Description	285
6.29.2	Constructor & Destructor Documentation	286
6.29.2.1	ActiveMQDestinationMarshaller	286
6.29.2.2	~ActiveMQDestinationMarshaller	286
6.29.3	Member Function Documentation	286
6.29.3.1	looseMarshal	286
6.29.3.2	looseUnmarshal	286
6.29.3.3	tightMarshal1	287
6.29.3.4	tightMarshal2	287
6.29.3.5	tightUnmarshal	288
6.30	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	
	Class Reference	289
6.30.1	Detailed Description	289

6.30.2	Constructor & Destructor Documentation	290
6.30.2.1	ActiveMQDestinationMarshaller	290
6.30.2.2	~ActiveMQDestinationMarshaller	290
6.30.3	Member Function Documentation	290
6.30.3.1	looseMarshal	290
6.30.3.2	looseUnmarshal	290
6.30.3.3	tightMarshal1	291
6.30.3.4	tightMarshal2	291
6.30.3.5	tightUnmarshal	292
6.31	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	
	Class Reference	293
6.31.1	Detailed Description	293
6.31.2	Constructor & Destructor Documentation	294
6.31.2.1	ActiveMQDestinationMarshaller	294
6.31.2.2	~ActiveMQDestinationMarshaller	294
6.31.3	Member Function Documentation	294
6.31.3.1	looseMarshal	294
6.31.3.2	looseUnmarshal	294
6.31.3.3	tightMarshal1	295
6.31.3.4	tightMarshal2	295
6.31.3.5	tightUnmarshal	296
6.32	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	
	Class Reference	297
6.32.1	Detailed Description	297
6.32.2	Constructor & Destructor Documentation	298
6.32.2.1	ActiveMQDestinationMarshaller	298
6.32.2.2	~ActiveMQDestinationMarshaller	298
6.32.3	Member Function Documentation	298
6.32.3.1	looseMarshal	298
6.32.3.2	looseUnmarshal	298
6.32.3.3	tightMarshal1	299
6.32.3.4	tightMarshal2	299
6.32.3.5	tightUnmarshal	300
6.33	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	
	Class Reference	301
6.33.1	Detailed Description	301
6.33.2	Constructor & Destructor Documentation	302

6.33.2.1	ActiveMQDestinationMarshaller	302
6.33.2.2	~ActiveMQDestinationMarshaller	302
6.33.3	Member Function Documentation	302
6.33.3.1	looseMarshal	302
6.33.3.2	looseUnmarshal	302
6.33.3.3	tightMarshal1	303
6.33.3.4	tightMarshal2	303
6.33.3.5	tightUnmarshal	304
6.34	activemq::exceptions::ActiveMQException Class Reference	305
6.34.1	Constructor & Destructor Documentation	305
6.34.1.1	ActiveMQException	305
6.34.1.2	ActiveMQException	305
6.34.1.3	ActiveMQException	306
6.34.1.4	ActiveMQException	306
6.34.1.5	~ActiveMQException	306
6.34.2	Member Function Documentation	306
6.34.2.1	clone	306
6.34.2.2	convertToCMSException	306
6.35	activemq::commands::ActiveMQMapMessage Class Reference	308
6.35.1	Constructor & Destructor Documentation	311
6.35.1.1	ActiveMQMapMessage	311
6.35.1.2	~ActiveMQMapMessage	311
6.35.2	Member Function Documentation	311
6.35.2.1	beforeMarshal	311
6.35.2.2	checkMapIsUnmarshalled	311
6.35.2.3	clearBody	311
6.35.2.4	clone	311
6.35.2.5	cloneDataStructure	312
6.35.2.6	copyDataStructure	312
6.35.2.7	equals	312
6.35.2.8	getBoolean	312
6.35.2.9	getByte	313
6.35.2.10	getBytes	313
6.35.2.11	getChar	313
6.35.2.12	getDataStructureType	314
6.35.2.13	getDouble	314

6.35.2.14	getFloat	314
6.35.2.15	getInt	314
6.35.2.16	getLong	315
6.35.2.17	getMap	315
6.35.2.18	getMap	315
6.35.2.19	getMapNames	315
6.35.2.20	getShort	316
6.35.2.21	getString	316
6.35.2.22	isMarshalAware	316
6.35.2.23	itemExists	316
6.35.2.24	setBoolean	317
6.35.2.25	setByte	317
6.35.2.26	setBytes	318
6.35.2.27	setChar	318
6.35.2.28	setDouble	318
6.35.2.29	setFloat	319
6.35.2.30	setInt	319
6.35.2.31	setLong	319
6.35.2.32	setShort	320
6.35.2.33	setString	320
6.35.2.34	toString	320
6.35.3	Field Documentation	320
6.35.3.1	ID_ACTIVEMQMAPMESSAGE	320
6.36	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	
	Class Reference	322
6.36.1	Detailed Description	322
6.36.2	Constructor & Destructor Documentation	323
6.36.2.1	ActiveMQMapMessageMarshaller	323
6.36.2.2	~ActiveMQMapMessageMarshaller	323
6.36.3	Member Function Documentation	323
6.36.3.1	createObject	323
6.36.3.2	getDataStructureType	323
6.36.3.3	looseMarshal	323
6.36.3.4	looseUnmarshal	324
6.36.3.5	tightMarshal1	324
6.36.3.6	tightMarshal2	324
6.36.3.7	tightUnmarshal	325

6.37	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	
	Class Reference	326
6.37.1	Detailed Description	326
6.37.2	Constructor & Destructor Documentation	327
	6.37.2.1 ActiveMQMapMessageMarshaller	327
	6.37.2.2 ~ActiveMQMapMessageMarshaller	327
6.37.3	Member Function Documentation	327
	6.37.3.1 createObject	327
	6.37.3.2 getDataStructureType	327
	6.37.3.3 looseMarshal	327
	6.37.3.4 looseUnmarshal	328
	6.37.3.5 tightMarshal1	328
	6.37.3.6 tightMarshal2	328
	6.37.3.7 tightUnmarshal	329
6.38	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	
	Class Reference	330
6.38.1	Detailed Description	330
6.38.2	Constructor & Destructor Documentation	331
	6.38.2.1 ActiveMQMapMessageMarshaller	331
	6.38.2.2 ~ActiveMQMapMessageMarshaller	331
6.38.3	Member Function Documentation	331
	6.38.3.1 createObject	331
	6.38.3.2 getDataStructureType	331
	6.38.3.3 looseMarshal	331
	6.38.3.4 looseUnmarshal	332
	6.38.3.5 tightMarshal1	332
	6.38.3.6 tightMarshal2	332
	6.38.3.7 tightUnmarshal	333
6.39	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	
	Class Reference	334
6.39.1	Detailed Description	334
6.39.2	Constructor & Destructor Documentation	335
	6.39.2.1 ActiveMQMapMessageMarshaller	335
	6.39.2.2 ~ActiveMQMapMessageMarshaller	335
6.39.3	Member Function Documentation	335
	6.39.3.1 createObject	335
	6.39.3.2 getDataStructureType	335

6.39.3.3	looseMarshal	335
6.39.3.4	looseUnmarshal	336
6.39.3.5	tightMarshal1	336
6.39.3.6	tightMarshal2	336
6.39.3.7	tightUnmarshal	337
6.40	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	
	Class Reference	338
6.40.1	Detailed Description	338
6.40.2	Constructor & Destructor Documentation	339
	6.40.2.1 ActiveMQMapMessageMarshaller	339
	6.40.2.2 ~ActiveMQMapMessageMarshaller	339
6.40.3	Member Function Documentation	339
	6.40.3.1 createObject	339
	6.40.3.2 getDataStructureType	339
	6.40.3.3 looseMarshal	339
	6.40.3.4 looseUnmarshal	340
	6.40.3.5 tightMarshal1	340
	6.40.3.6 tightMarshal2	340
	6.40.3.7 tightUnmarshal	341
6.41	activemq::commands::ActiveMQMessage Class Reference	342
	6.41.1 Constructor & Destructor Documentation	342
	6.41.1.1 ActiveMQMessage	342
	6.41.1.2 ~ActiveMQMessage	342
6.41.2	Member Function Documentation	342
	6.41.2.1 clone	342
	6.41.2.2 cloneDataStructure	343
	6.41.2.3 copyDataStructure	343
	6.41.2.4 equals	343
	6.41.2.5 getDataStructureType	343
	6.41.2.6 toString	344
6.41.3	Field Documentation	344
	6.41.3.1 ID_ACTIVEMQMESSAGE	344
6.42	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class	
	Reference	345
6.42.1	Detailed Description	345
6.42.2	Constructor & Destructor Documentation	346
	6.42.2.1 ActiveMQMessageMarshaller	346

6.42.2.2	<code>~ActiveMQMessageMarshaller</code>	346
6.42.3	Member Function Documentation	346
6.42.3.1	<code>createObject</code>	346
6.42.3.2	<code>getDataStructureType</code>	346
6.42.3.3	<code>looseMarshal</code>	346
6.42.3.4	<code>looseUnmarshal</code>	347
6.42.3.5	<code>tightMarshal1</code>	347
6.42.3.6	<code>tightMarshal2</code>	347
6.42.3.7	<code>tightUnmarshal</code>	348
6.43	<code>activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller</code> Class Reference	349
6.43.1	Detailed Description	349
6.43.2	Constructor & Destructor Documentation	350
6.43.2.1	<code>ActiveMQMessageMarshaller</code>	350
6.43.2.2	<code>~ActiveMQMessageMarshaller</code>	350
6.43.3	Member Function Documentation	350
6.43.3.1	<code>createObject</code>	350
6.43.3.2	<code>getDataStructureType</code>	350
6.43.3.3	<code>looseMarshal</code>	350
6.43.3.4	<code>looseUnmarshal</code>	351
6.43.3.5	<code>tightMarshal1</code>	351
6.43.3.6	<code>tightMarshal2</code>	351
6.43.3.7	<code>tightUnmarshal</code>	352
6.44	<code>activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller</code> Class Reference	353
6.44.1	Detailed Description	353
6.44.2	Constructor & Destructor Documentation	354
6.44.2.1	<code>ActiveMQMessageMarshaller</code>	354
6.44.2.2	<code>~ActiveMQMessageMarshaller</code>	354
6.44.3	Member Function Documentation	354
6.44.3.1	<code>createObject</code>	354
6.44.3.2	<code>getDataStructureType</code>	354
6.44.3.3	<code>looseMarshal</code>	354
6.44.3.4	<code>looseUnmarshal</code>	355
6.44.3.5	<code>tightMarshal1</code>	355
6.44.3.6	<code>tightMarshal2</code>	355
6.44.3.7	<code>tightUnmarshal</code>	356

6.45	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	Class	
	Reference		357
6.45.1	Detailed Description		357
6.45.2	Constructor & Destructor Documentation		358
6.45.2.1	ActiveMQMessageMarshaller		358
6.45.2.2	~ActiveMQMessageMarshaller		358
6.45.3	Member Function Documentation		358
6.45.3.1	createObject		358
6.45.3.2	getDataStructureType		358
6.45.3.3	looseMarshal		358
6.45.3.4	looseUnmarshal		359
6.45.3.5	tightMarshal1		359
6.45.3.6	tightMarshal2		359
6.45.3.7	tightUnmarshal		360
6.46	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	Class	
	Reference		361
6.46.1	Detailed Description		361
6.46.2	Constructor & Destructor Documentation		362
6.46.2.1	ActiveMQMessageMarshaller		362
6.46.2.2	~ActiveMQMessageMarshaller		362
6.46.3	Member Function Documentation		362
6.46.3.1	createObject		362
6.46.3.2	getDataStructureType		362
6.46.3.3	looseMarshal		362
6.46.3.4	looseUnmarshal		363
6.46.3.5	tightMarshal1		363
6.46.3.6	tightMarshal2		363
6.46.3.7	tightUnmarshal		364
6.47	activemq::commands::ActiveMQMessageTemplate< T >	Class Template Reference	365
6.47.1	Constructor & Destructor Documentation		368
6.47.1.1	ActiveMQMessageTemplate		368
6.47.1.2	~ActiveMQMessageTemplate		368
6.47.2	Member Function Documentation		368
6.47.2.1	acknowledge		368
6.47.2.2	clearBody		368
6.47.2.3	clearProperties		369
6.47.2.4	equals		369

6.47.2.5	failIfReadOnlyBody	369
6.47.2.6	failIfReadOnlyProperties	369
6.47.2.7	failIfWriteOnlyBody	369
6.47.2.8	getBooleanProperty	369
6.47.2.9	getByteProperty	370
6.47.2.10	getCMSCorrelationID	370
6.47.2.11	getCMSDeliveryMode	370
6.47.2.12	getCMSDestination	371
6.47.2.13	getCMSExpiration	371
6.47.2.14	getCMSMessageID	371
6.47.2.15	getCMSPriority	372
6.47.2.16	getCMSRedelivered	372
6.47.2.17	getCMSReplyTo	372
6.47.2.18	getCMSTimestamp	372
6.47.2.19	getCMSType	373
6.47.2.20	getDoubleProperty	373
6.47.2.21	getFloatProperty	373
6.47.2.22	getIntProperty	374
6.47.2.23	getLongProperty	374
6.47.2.24	getPropertyNames	375
6.47.2.25	getShortProperty	375
6.47.2.26	getStringProperty	375
6.47.2.27	onSend	376
6.47.2.28	propertyExists	376
6.47.2.29	setBooleanProperty	376
6.47.2.30	setByteProperty	376
6.47.2.31	setCMSCorrelationID	377
6.47.2.32	setCMSDeliveryMode	377
6.47.2.33	setCMSDestination	377
6.47.2.34	setCMSExpiration	378
6.47.2.35	setCMSMessageID	378
6.47.2.36	setCMSPriority	378
6.47.2.37	setCMSRedelivered	378
6.47.2.38	setCMSReplyTo	379
6.47.2.39	setCMSTimestamp	379
6.47.2.40	setCMSType	379

6.47.2.41	setDoubleProperty	380
6.47.2.42	setFloatProperty	380
6.47.2.43	setIntProperty	380
6.47.2.44	setLongProperty	381
6.47.2.45	setShortProperty	381
6.47.2.46	setStringProperty	381
6.48	activemq::commands::ActiveMQObjectMessage Class Reference	383
6.48.1	Constructor & Destructor Documentation	384
6.48.1.1	ActiveMQObjectMessage	384
6.48.1.2	~ActiveMQObjectMessage	384
6.48.2	Member Function Documentation	384
6.48.2.1	clone	384
6.48.2.2	cloneDataStructure	384
6.48.2.3	copyDataStructure	384
6.48.2.4	equals	384
6.48.2.5	getDataStructureType	385
6.48.2.6	toString	385
6.48.3	Field Documentation	385
6.48.3.1	ID_ACTIVEMQOBJECTMESSAGE	385
6.49	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference	386
6.49.1	Detailed Description	386
6.49.2	Constructor & Destructor Documentation	387
6.49.2.1	ActiveMQObjectMessageMarshaller	387
6.49.2.2	~ActiveMQObjectMessageMarshaller	387
6.49.3	Member Function Documentation	387
6.49.3.1	createObject	387
6.49.3.2	getDataStructureType	387
6.49.3.3	looseMarshal	387
6.49.3.4	looseUnmarshal	388
6.49.3.5	tightMarshal1	388
6.49.3.6	tightMarshal2	388
6.49.3.7	tightUnmarshal	389
6.50	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference	390
6.50.1	Detailed Description	390
6.50.2	Constructor & Destructor Documentation	391

6.50.2.1	ActiveMQObjectMessageMarshaller	391
6.50.2.2	~ActiveMQObjectMessageMarshaller	391
6.50.3	Member Function Documentation	391
6.50.3.1	createObject	391
6.50.3.2	getDataStructureType	391
6.50.3.3	looseMarshal	391
6.50.3.4	looseUnmarshal	392
6.50.3.5	tightMarshal1	392
6.50.3.6	tightMarshal2	392
6.50.3.7	tightUnmarshal	393
6.51	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	
	Class Reference	394
6.51.1	Detailed Description	394
6.51.2	Constructor & Destructor Documentation	395
6.51.2.1	ActiveMQObjectMessageMarshaller	395
6.51.2.2	~ActiveMQObjectMessageMarshaller	395
6.51.3	Member Function Documentation	395
6.51.3.1	createObject	395
6.51.3.2	getDataStructureType	395
6.51.3.3	looseMarshal	395
6.51.3.4	looseUnmarshal	396
6.51.3.5	tightMarshal1	396
6.51.3.6	tightMarshal2	396
6.51.3.7	tightUnmarshal	397
6.52	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	
	Class Reference	398
6.52.1	Detailed Description	398
6.52.2	Constructor & Destructor Documentation	399
6.52.2.1	ActiveMQObjectMessageMarshaller	399
6.52.2.2	~ActiveMQObjectMessageMarshaller	399
6.52.3	Member Function Documentation	399
6.52.3.1	createObject	399
6.52.3.2	getDataStructureType	399
6.52.3.3	looseMarshal	399
6.52.3.4	looseUnmarshal	400
6.52.3.5	tightMarshal1	400
6.52.3.6	tightMarshal2	400

6.52.3.7	tightUnmarshal	401
6.53	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	
	Class Reference	402
6.53.1	Detailed Description	402
6.53.2	Constructor & Destructor Documentation	403
6.53.2.1	ActiveMQObjectMessageMarshaller	403
6.53.2.2	~ActiveMQObjectMessageMarshaller	403
6.53.3	Member Function Documentation	403
6.53.3.1	createObject	403
6.53.3.2	getDataStructureType	403
6.53.3.3	looseMarshal	403
6.53.3.4	looseUnmarshal	404
6.53.3.5	tightMarshal1	404
6.53.3.6	tightMarshal2	404
6.53.3.7	tightUnmarshal	405
6.54	activemq::core::ActiveMQProducer Class Reference	406
6.54.1	Constructor & Destructor Documentation	407
6.54.1.1	ActiveMQProducer	407
6.54.1.2	~ActiveMQProducer	408
6.54.2	Member Function Documentation	408
6.54.2.1	close	408
6.54.2.2	getDeliveryMode	408
6.54.2.3	getDisableMessageID	408
6.54.2.4	getDisableMessageTimeStamp	408
6.54.2.5	getPriority	409
6.54.2.6	getProducerId	409
6.54.2.7	getProducerInfo	409
6.54.2.8	getSendTimeout	409
6.54.2.9	getTimeToLive	409
6.54.2.10	isClosed	410
6.54.2.11	onProducerAck	410
6.54.2.12	send	410
6.54.2.13	send	410
6.54.2.14	send	411
6.54.2.15	send	411
6.54.2.16	setDeliveryMode	412
6.54.2.17	setDisableMessageID	412

6.54.2.18	setDisableMessageTimeStamp	412
6.54.2.19	setPriority	412
6.54.2.20	setSendTimeout	413
6.54.2.21	setTimeToLive	413
6.55	activemq::util::ActiveMQProperties Class Reference	414
6.55.1	Detailed Description	415
6.55.2	Constructor & Destructor Documentation	415
6.55.2.1	~ActiveMQProperties	415
6.55.3	Member Function Documentation	415
6.55.3.1	clear	415
6.55.3.2	clone	415
6.55.3.3	copy	415
6.55.3.4	getProperties	416
6.55.3.5	getProperties	416
6.55.3.6	getProperty	416
6.55.3.7	getProperty	416
6.55.3.8	hasProperty	416
6.55.3.9	isEmpty	417
6.55.3.10	remove	417
6.55.3.11	setProperties	417
6.55.3.12	setProperty	417
6.55.3.13	toArray	417
6.55.3.14	toString	417
6.56	activemq::commands::ActiveMQQueue Class Reference	419
6.56.1	Constructor & Destructor Documentation	420
6.56.1.1	ActiveMQQueue	420
6.56.1.2	ActiveMQQueue	420
6.56.1.3	~ActiveMQQueue	420
6.56.2	Member Function Documentation	420
6.56.2.1	clone	420
6.56.2.2	cloneDataStructure	420
6.56.2.3	copy	420
6.56.2.4	copyDataStructure	420
6.56.2.5	equals	421
6.56.2.6	getCMSDestination	421
6.56.2.7	getCMSProperties	421

6.56.2.8	getDataStructureType	421
6.56.2.9	getDestinationType	422
6.56.2.10	getQueueName	422
6.56.2.11	toString	422
6.56.3	Field Documentation	422
6.56.3.1	ID_ACTIVEMQUEUE	422
6.57	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class	
	Reference	423
6.57.1	Detailed Description	423
6.57.2	Constructor & Destructor Documentation	424
6.57.2.1	ActiveMQQueueMarshaller	424
6.57.2.2	~ActiveMQQueueMarshaller	424
6.57.3	Member Function Documentation	424
6.57.3.1	createObject	424
6.57.3.2	getDataStructureType	424
6.57.3.3	looseMarshal	424
6.57.3.4	looseUnmarshal	425
6.57.3.5	tightMarshal1	425
6.57.3.6	tightMarshal2	425
6.57.3.7	tightUnmarshal	426
6.58	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class	
	Reference	427
6.58.1	Detailed Description	427
6.58.2	Constructor & Destructor Documentation	428
6.58.2.1	ActiveMQQueueMarshaller	428
6.58.2.2	~ActiveMQQueueMarshaller	428
6.58.3	Member Function Documentation	428
6.58.3.1	createObject	428
6.58.3.2	getDataStructureType	428
6.58.3.3	looseMarshal	428
6.58.3.4	looseUnmarshal	429
6.58.3.5	tightMarshal1	429
6.58.3.6	tightMarshal2	429
6.58.3.7	tightUnmarshal	430
6.59	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller Class	
	Reference	431
6.59.1	Detailed Description	431

6.59.2	Constructor & Destructor Documentation	432
6.59.2.1	ActiveMQQueueMarshaller	432
6.59.2.2	~ActiveMQQueueMarshaller	432
6.59.3	Member Function Documentation	432
6.59.3.1	createObject	432
6.59.3.2	getDataStructureType	432
6.59.3.3	looseMarshal	432
6.59.3.4	looseUnmarshal	433
6.59.3.5	tightMarshal1	433
6.59.3.6	tightMarshal2	433
6.59.3.7	tightUnmarshal	434
6.60	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller Class Reference	435
6.60.1	Detailed Description	435
6.60.2	Constructor & Destructor Documentation	436
6.60.2.1	ActiveMQQueueMarshaller	436
6.60.2.2	~ActiveMQQueueMarshaller	436
6.60.3	Member Function Documentation	436
6.60.3.1	createObject	436
6.60.3.2	getDataStructureType	436
6.60.3.3	looseMarshal	436
6.60.3.4	looseUnmarshal	437
6.60.3.5	tightMarshal1	437
6.60.3.6	tightMarshal2	437
6.60.3.7	tightUnmarshal	438
6.61	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller Class Reference	439
6.61.1	Detailed Description	439
6.61.2	Constructor & Destructor Documentation	440
6.61.2.1	ActiveMQQueueMarshaller	440
6.61.2.2	~ActiveMQQueueMarshaller	440
6.61.3	Member Function Documentation	440
6.61.3.1	createObject	440
6.61.3.2	getDataStructureType	440
6.61.3.3	looseMarshal	440
6.61.3.4	looseUnmarshal	441
6.61.3.5	tightMarshal1	441

6.61.3.6	tightMarshal2	441
6.61.3.7	tightUnmarshal	442
6.62	activemq::core::ActiveMQSession Class Reference	443
6.62.1	Constructor & Destructor Documentation	447
6.62.1.1	ActiveMQSession	447
6.62.1.2	~ActiveMQSession	447
6.62.2	Member Function Documentation	447
6.62.2.1	acknowledge	447
6.62.2.2	clearMessagesInProgress	447
6.62.2.3	close	447
6.62.2.4	commit	447
6.62.2.5	createBrowser	447
6.62.2.6	createBrowser	448
6.62.2.7	createBytesMessage	448
6.62.2.8	createBytesMessage	449
6.62.2.9	createConsumer	449
6.62.2.10	createConsumer	449
6.62.2.11	createConsumer	450
6.62.2.12	createDurableConsumer	450
6.62.2.13	createMapMessage	450
6.62.2.14	createMessage	451
6.62.2.15	createProducer	451
6.62.2.16	createQueue	451
6.62.2.17	createStreamMessage	451
6.62.2.18	createTemporaryQueue	452
6.62.2.19	createTemporaryTopic	452
6.62.2.20	createTextMessage	452
6.62.2.21	createTextMessage	452
6.62.2.22	createTopic	453
6.62.2.23	deliverAcks	453
6.62.2.24	dispatch	453
6.62.2.25	disposeOf	453
6.62.2.26	disposeOf	454
6.62.2.27	doStartTransaction	454
6.62.2.28	fire	454
6.62.2.29	getAcknowledgeMode	454

6.62.2.30	getConnection	454
6.62.2.31	getExceptionListener	455
6.62.2.32	getLastDeliveredSequenceId	455
6.62.2.33	getSessionId	455
6.62.2.34	getSessionInfo	455
6.62.2.35	isAutoAcknowledge	455
6.62.2.36	isClientAcknowledge	455
6.62.2.37	isDupsOkAcknowledge	456
6.62.2.38	isIndividualAcknowledge	456
6.62.2.39	isStarted	456
6.62.2.40	isTransacted	456
6.62.2.41	oneway	456
6.62.2.42	recover	456
6.62.2.43	redispach	457
6.62.2.44	rollback	457
6.62.2.45	send	457
6.62.2.46	setLastDeliveredSequenceId	458
6.62.2.47	start	458
6.62.2.48	stop	458
6.62.2.49	syncRequest	458
6.62.2.50	unsubscribe	458
6.62.2.51	wakeup	459
6.62.3	Friends And Related Function Documentation	459
6.62.3.1	ActiveMQSessionExecutor	459
6.63	activemq::core::ActiveMQSessionExecutor Class Reference	460
6.63.1	Detailed Description	461
6.63.2	Constructor & Destructor Documentation	461
6.63.2.1	ActiveMQSessionExecutor	461
6.63.2.2	~ActiveMQSessionExecutor	461
6.63.3	Member Function Documentation	461
6.63.3.1	clear	461
6.63.3.2	clearMessagesInProgress	461
6.63.3.3	close	461
6.63.3.4	execute	461
6.63.3.5	executeFirst	461
6.63.3.6	getUnconsumedMessages	462

6.63.3.7	hasUnconsumedMessages	462
6.63.3.8	isEmpty	462
6.63.3.9	isRunning	462
6.63.3.10	iterate	462
6.63.3.11	start	462
6.63.3.12	stop	463
6.63.3.13	wakeup	463
6.64	activemq::commands::ActiveMQStreamMessage Class Reference	464
6.64.1	Constructor & Destructor Documentation	467
6.64.1.1	ActiveMQStreamMessage	467
6.64.1.2	~ActiveMQStreamMessage	467
6.64.2	Member Function Documentation	467
6.64.2.1	clearBody	467
6.64.2.2	clone	467
6.64.2.3	cloneDataStructure	467
6.64.2.4	copyDataStructure	467
6.64.2.5	equals	468
6.64.2.6	getDataStructureType	468
6.64.2.7	onSend	468
6.64.2.8	readBoolean	468
6.64.2.9	readByte	469
6.64.2.10	readBytes	469
6.64.2.11	readBytes	470
6.64.2.12	readChar	470
6.64.2.13	readDouble	471
6.64.2.14	readFloat	471
6.64.2.15	readInt	472
6.64.2.16	readLong	472
6.64.2.17	readShort	472
6.64.2.18	readString	473
6.64.2.19	readUnsignedShort	473
6.64.2.20	reset	474
6.64.2.21	toString	474
6.64.2.22	writeBoolean	474
6.64.2.23	writeByte	474
6.64.2.24	writeBytes	475

6.64.2.25	writeBytes	475
6.64.2.26	writeChar	475
6.64.2.27	writeDouble	476
6.64.2.28	writeFloat	476
6.64.2.29	writeInt	476
6.64.2.30	writeLong	477
6.64.2.31	writeShort	477
6.64.2.32	writeString	477
6.64.2.33	writeUnsignedShort	478
6.64.3	Field Documentation	478
6.64.3.1	ID_ACTIVEMQSTREAMMESSAGE	478
6.65	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	
	Class Reference	479
6.65.1	Detailed Description	479
6.65.2	Constructor & Destructor Documentation	480
6.65.2.1	ActiveMQStreamMessageMarshaller	480
6.65.2.2	~ActiveMQStreamMessageMarshaller	480
6.65.3	Member Function Documentation	480
6.65.3.1	createObject	480
6.65.3.2	getDataStructureType	480
6.65.3.3	looseMarshal	480
6.65.3.4	looseUnmarshal	481
6.65.3.5	tightMarshal1	481
6.65.3.6	tightMarshal2	481
6.65.3.7	tightUnmarshal	482
6.66	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	
	Class Reference	483
6.66.1	Detailed Description	483
6.66.2	Constructor & Destructor Documentation	484
6.66.2.1	ActiveMQStreamMessageMarshaller	484
6.66.2.2	~ActiveMQStreamMessageMarshaller	484
6.66.3	Member Function Documentation	484
6.66.3.1	createObject	484
6.66.3.2	getDataStructureType	484
6.66.3.3	looseMarshal	484
6.66.3.4	looseUnmarshal	485
6.66.3.5	tightMarshal1	485

6.66.3.6	tightMarshal2	485
6.66.3.7	tightUnmarshal	486
6.67	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	
	Class Reference	487
6.67.1	Detailed Description	487
6.67.2	Constructor & Destructor Documentation	488
6.67.2.1	ActiveMQStreamMessageMarshaller	488
6.67.2.2	~ActiveMQStreamMessageMarshaller	488
6.67.3	Member Function Documentation	488
6.67.3.1	createObject	488
6.67.3.2	getDataStructureType	488
6.67.3.3	looseMarshal	488
6.67.3.4	looseUnmarshal	489
6.67.3.5	tightMarshal1	489
6.67.3.6	tightMarshal2	489
6.67.3.7	tightUnmarshal	490
6.68	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	
	Class Reference	491
6.68.1	Detailed Description	491
6.68.2	Constructor & Destructor Documentation	492
6.68.2.1	ActiveMQStreamMessageMarshaller	492
6.68.2.2	~ActiveMQStreamMessageMarshaller	492
6.68.3	Member Function Documentation	492
6.68.3.1	createObject	492
6.68.3.2	getDataStructureType	492
6.68.3.3	looseMarshal	492
6.68.3.4	looseUnmarshal	493
6.68.3.5	tightMarshal1	493
6.68.3.6	tightMarshal2	493
6.68.3.7	tightUnmarshal	494
6.69	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	
	Class Reference	495
6.69.1	Detailed Description	495
6.69.2	Constructor & Destructor Documentation	496
6.69.2.1	ActiveMQStreamMessageMarshaller	496
6.69.2.2	~ActiveMQStreamMessageMarshaller	496
6.69.3	Member Function Documentation	496

6.69.3.1	createObject	496
6.69.3.2	getDataStructureType	496
6.69.3.3	looseMarshal	496
6.69.3.4	looseUnmarshal	497
6.69.3.5	tightMarshal1	497
6.69.3.6	tightMarshal2	497
6.69.3.7	tightUnmarshal	498
6.70	activemq::commands::ActiveMQTempDestination Class Reference	499
6.70.1	Constructor & Destructor Documentation	500
6.70.1.1	ActiveMQTempDestination	500
6.70.1.2	ActiveMQTempDestination	500
6.70.1.3	~ActiveMQTempDestination	500
6.70.2	Member Function Documentation	500
6.70.2.1	cloneDataStructure	500
6.70.2.2	close	500
6.70.2.3	copyDataStructure	500
6.70.2.4	equals	501
6.70.2.5	getDataStructureType	501
6.70.2.6	setConnection	501
6.70.2.7	toString	501
6.70.3	Field Documentation	502
6.70.3.1	connection	502
6.70.3.2	ID_ACTIVEMQTEMPDESTINATION	502
6.71	activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller Class Reference	503
6.71.1	Detailed Description	503
6.71.2	Constructor & Destructor Documentation	504
6.71.2.1	ActiveMQTempDestinationMarshaller	504
6.71.2.2	~ActiveMQTempDestinationMarshaller	504
6.71.3	Member Function Documentation	504
6.71.3.1	looseMarshal	504
6.71.3.2	looseUnmarshal	504
6.71.3.3	tightMarshal1	505
6.71.3.4	tightMarshal2	505
6.71.3.5	tightUnmarshal	506
6.72	activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller Class Reference	507

6.72.1	Detailed Description	507
6.72.2	Constructor & Destructor Documentation	508
6.72.2.1	ActiveMQTempDestinationMarshaller	508
6.72.2.2	~ActiveMQTempDestinationMarshaller	508
6.72.3	Member Function Documentation	508
6.72.3.1	looseMarshal	508
6.72.3.2	looseUnmarshal	508
6.72.3.3	tightMarshal1	509
6.72.3.4	tightMarshal2	509
6.72.3.5	tightUnmarshal	510
6.73	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller Class Reference	511
6.73.1	Detailed Description	511
6.73.2	Constructor & Destructor Documentation	512
6.73.2.1	ActiveMQTempDestinationMarshaller	512
6.73.2.2	~ActiveMQTempDestinationMarshaller	512
6.73.3	Member Function Documentation	512
6.73.3.1	looseMarshal	512
6.73.3.2	looseUnmarshal	512
6.73.3.3	tightMarshal1	513
6.73.3.4	tightMarshal2	513
6.73.3.5	tightUnmarshal	514
6.74	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference	515
6.74.1	Detailed Description	515
6.74.2	Constructor & Destructor Documentation	516
6.74.2.1	ActiveMQTempDestinationMarshaller	516
6.74.2.2	~ActiveMQTempDestinationMarshaller	516
6.74.3	Member Function Documentation	516
6.74.3.1	looseMarshal	516
6.74.3.2	looseUnmarshal	516
6.74.3.3	tightMarshal1	517
6.74.3.4	tightMarshal2	517
6.74.3.5	tightUnmarshal	518
6.75	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller Class Reference	519
6.75.1	Detailed Description	519

6.75.2	Constructor & Destructor Documentation	520
6.75.2.1	ActiveMQTempDestinationMarshaller	520
6.75.2.2	~ActiveMQTempDestinationMarshaller	520
6.75.3	Member Function Documentation	520
6.75.3.1	looseMarshal	520
6.75.3.2	looseUnmarshal	520
6.75.3.3	tightMarshal1	521
6.75.3.4	tightMarshal2	521
6.75.3.5	tightUnmarshal	522
6.76	activemq::commands::ActiveMQTempQueue Class Reference	523
6.76.1	Constructor & Destructor Documentation	524
6.76.1.1	ActiveMQTempQueue	524
6.76.1.2	ActiveMQTempQueue	524
6.76.1.3	~ActiveMQTempQueue	524
6.76.2	Member Function Documentation	524
6.76.2.1	clone	524
6.76.2.2	cloneDataStructure	524
6.76.2.3	copy	524
6.76.2.4	copyDataStructure	525
6.76.2.5	destroy	525
6.76.2.6	equals	525
6.76.2.7	getCMSDestination	525
6.76.2.8	getCMSProperties	525
6.76.2.9	getDataStructureType	526
6.76.2.10	getDestinationType	526
6.76.2.11	getQueueName	526
6.76.2.12	toString	526
6.76.3	Field Documentation	527
6.76.3.1	ID_ACTIVEMQTEMPQUEUE	527
6.77	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller Class Reference	528
6.77.1	Detailed Description	528
6.77.2	Constructor & Destructor Documentation	529
6.77.2.1	ActiveMQTempQueueMarshaller	529
6.77.2.2	~ActiveMQTempQueueMarshaller	529
6.77.3	Member Function Documentation	529
6.77.3.1	createObject	529

6.77.3.2	getDataStructureType	529
6.77.3.3	looseMarshal	529
6.77.3.4	looseUnmarshal	530
6.77.3.5	tightMarshal1	530
6.77.3.6	tightMarshal2	530
6.77.3.7	tightUnmarshal	531
6.78	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	
	Class Reference	532
6.78.1	Detailed Description	532
6.78.2	Constructor & Destructor Documentation	533
6.78.2.1	ActiveMQTempQueueMarshaller	533
6.78.2.2	~ActiveMQTempQueueMarshaller	533
6.78.3	Member Function Documentation	533
6.78.3.1	createObject	533
6.78.3.2	getDataStructureType	533
6.78.3.3	looseMarshal	533
6.78.3.4	looseUnmarshal	534
6.78.3.5	tightMarshal1	534
6.78.3.6	tightMarshal2	534
6.78.3.7	tightUnmarshal	535
6.79	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	
	Class Reference	536
6.79.1	Detailed Description	536
6.79.2	Constructor & Destructor Documentation	537
6.79.2.1	ActiveMQTempQueueMarshaller	537
6.79.2.2	~ActiveMQTempQueueMarshaller	537
6.79.3	Member Function Documentation	537
6.79.3.1	createObject	537
6.79.3.2	getDataStructureType	537
6.79.3.3	looseMarshal	537
6.79.3.4	looseUnmarshal	538
6.79.3.5	tightMarshal1	538
6.79.3.6	tightMarshal2	538
6.79.3.7	tightUnmarshal	539
6.80	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	
	Class Reference	540
6.80.1	Detailed Description	540

6.80.2	Constructor & Destructor Documentation	541
6.80.2.1	ActiveMQTempQueueMarshaller	541
6.80.2.2	~ActiveMQTempQueueMarshaller	541
6.80.3	Member Function Documentation	541
6.80.3.1	createObject	541
6.80.3.2	getDataStructureType	541
6.80.3.3	looseMarshal	541
6.80.3.4	looseUnmarshal	542
6.80.3.5	tightMarshal1	542
6.80.3.6	tightMarshal2	542
6.80.3.7	tightUnmarshal	543
6.81	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference	544
6.81.1	Detailed Description	544
6.81.2	Constructor & Destructor Documentation	545
6.81.2.1	ActiveMQTempQueueMarshaller	545
6.81.2.2	~ActiveMQTempQueueMarshaller	545
6.81.3	Member Function Documentation	545
6.81.3.1	createObject	545
6.81.3.2	getDataStructureType	545
6.81.3.3	looseMarshal	545
6.81.3.4	looseUnmarshal	546
6.81.3.5	tightMarshal1	546
6.81.3.6	tightMarshal2	546
6.81.3.7	tightUnmarshal	547
6.82	activemq::commands::ActiveMQTempTopic Class Reference	548
6.82.1	Constructor & Destructor Documentation	549
6.82.1.1	ActiveMQTempTopic	549
6.82.1.2	ActiveMQTempTopic	549
6.82.1.3	~ActiveMQTempTopic	549
6.82.2	Member Function Documentation	549
6.82.2.1	clone	549
6.82.2.2	cloneDataStructure	549
6.82.2.3	copy	549
6.82.2.4	copyDataStructure	550
6.82.2.5	destroy	550
6.82.2.6	equals	550

6.82.2.7	getCMSDestination	550
6.82.2.8	getCMSProperties	550
6.82.2.9	getDataStructureType	551
6.82.2.10	getDestinationType	551
6.82.2.11	getTopicName	551
6.82.2.12	toString	551
6.82.3	Field Documentation	552
6.82.3.1	ID_ACTIVEMQTEMPTOPIC	552
6.83	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	
	Class Reference	553
6.83.1	Detailed Description	553
6.83.2	Constructor & Destructor Documentation	554
6.83.2.1	ActiveMQTempTopicMarshaller	554
6.83.2.2	~ActiveMQTempTopicMarshaller	554
6.83.3	Member Function Documentation	554
6.83.3.1	createObject	554
6.83.3.2	getDataStructureType	554
6.83.3.3	looseMarshal	554
6.83.3.4	looseUnmarshal	555
6.83.3.5	tightMarshal1	555
6.83.3.6	tightMarshal2	555
6.83.3.7	tightUnmarshal	556
6.84	activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	
	Class Reference	557
6.84.1	Detailed Description	557
6.84.2	Constructor & Destructor Documentation	558
6.84.2.1	ActiveMQTempTopicMarshaller	558
6.84.2.2	~ActiveMQTempTopicMarshaller	558
6.84.3	Member Function Documentation	558
6.84.3.1	createObject	558
6.84.3.2	getDataStructureType	558
6.84.3.3	looseMarshal	558
6.84.3.4	looseUnmarshal	559
6.84.3.5	tightMarshal1	559
6.84.3.6	tightMarshal2	559
6.84.3.7	tightUnmarshal	560

6.85	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	
	Class Reference	561
6.85.1	Detailed Description	561
6.85.2	Constructor & Destructor Documentation	562
	6.85.2.1 ActiveMQTempTopicMarshaller	562
	6.85.2.2 ~ActiveMQTempTopicMarshaller	562
6.85.3	Member Function Documentation	562
	6.85.3.1 createObject	562
	6.85.3.2 getDataStructureType	562
	6.85.3.3 looseMarshal	562
	6.85.3.4 looseUnmarshal	563
	6.85.3.5 tightMarshal1	563
	6.85.3.6 tightMarshal2	563
	6.85.3.7 tightUnmarshal	564
6.86	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	
	Class Reference	565
6.86.1	Detailed Description	565
6.86.2	Constructor & Destructor Documentation	566
	6.86.2.1 ActiveMQTempTopicMarshaller	566
	6.86.2.2 ~ActiveMQTempTopicMarshaller	566
6.86.3	Member Function Documentation	566
	6.86.3.1 createObject	566
	6.86.3.2 getDataStructureType	566
	6.86.3.3 looseMarshal	566
	6.86.3.4 looseUnmarshal	567
	6.86.3.5 tightMarshal1	567
	6.86.3.6 tightMarshal2	567
	6.86.3.7 tightUnmarshal	568
6.87	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	
	Class Reference	569
6.87.1	Detailed Description	569
6.87.2	Constructor & Destructor Documentation	570
	6.87.2.1 ActiveMQTempTopicMarshaller	570
	6.87.2.2 ~ActiveMQTempTopicMarshaller	570
6.87.3	Member Function Documentation	570
	6.87.3.1 createObject	570
	6.87.3.2 getDataStructureType	570

6.87.3.3	looseMarshal	570
6.87.3.4	looseUnmarshal	571
6.87.3.5	tightMarshal1	571
6.87.3.6	tightMarshal2	571
6.87.3.7	tightUnmarshal	572
6.88	activemq::commands::ActiveMQTextMessage Class Reference	573
6.88.1	Constructor & Destructor Documentation	574
6.88.1.1	ActiveMQTextMessage	574
6.88.1.2	~ActiveMQTextMessage	574
6.88.2	Member Function Documentation	574
6.88.2.1	beforeMarshal	574
6.88.2.2	clearBody	574
6.88.2.3	clone	574
6.88.2.4	cloneDataStructure	575
6.88.2.5	copyDataStructure	575
6.88.2.6	equals	575
6.88.2.7	getDataStructureType	575
6.88.2.8	getSize	576
6.88.2.9	getText	576
6.88.2.10	setText	576
6.88.2.11	setText	576
6.88.2.12	toString	577
6.88.3	Field Documentation	577
6.88.3.1	ID_ACTIVEMQTEXTMESSAGE	577
6.88.3.2	text	577
6.89	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference	578
6.89.1	Detailed Description	578
6.89.2	Constructor & Destructor Documentation	579
6.89.2.1	ActiveMQTextMessageMarshaller	579
6.89.2.2	~ActiveMQTextMessageMarshaller	579
6.89.3	Member Function Documentation	579
6.89.3.1	createObject	579
6.89.3.2	getDataStructureType	579
6.89.3.3	looseMarshal	579
6.89.3.4	looseUnmarshal	580
6.89.3.5	tightMarshal1	580

6.89.3.6	tightMarshal2	580
6.89.3.7	tightUnmarshal	581
6.90	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	
	Class Reference	582
6.90.1	Detailed Description	582
6.90.2	Constructor & Destructor Documentation	583
6.90.2.1	ActiveMQTextMessageMarshaller	583
6.90.2.2	~ActiveMQTextMessageMarshaller	583
6.90.3	Member Function Documentation	583
6.90.3.1	createObject	583
6.90.3.2	getDataStructureType	583
6.90.3.3	looseMarshal	583
6.90.3.4	looseUnmarshal	584
6.90.3.5	tightMarshal1	584
6.90.3.6	tightMarshal2	584
6.90.3.7	tightUnmarshal	585
6.91	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	
	Class Reference	586
6.91.1	Detailed Description	586
6.91.2	Constructor & Destructor Documentation	587
6.91.2.1	ActiveMQTextMessageMarshaller	587
6.91.2.2	~ActiveMQTextMessageMarshaller	587
6.91.3	Member Function Documentation	587
6.91.3.1	createObject	587
6.91.3.2	getDataStructureType	587
6.91.3.3	looseMarshal	587
6.91.3.4	looseUnmarshal	588
6.91.3.5	tightMarshal1	588
6.91.3.6	tightMarshal2	588
6.91.3.7	tightUnmarshal	589
6.92	activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	
	Class Reference	590
6.92.1	Detailed Description	590
6.92.2	Constructor & Destructor Documentation	591
6.92.2.1	ActiveMQTextMessageMarshaller	591
6.92.2.2	~ActiveMQTextMessageMarshaller	591
6.92.3	Member Function Documentation	591

6.92.3.1	createObject	591
6.92.3.2	getDataStructureType	591
6.92.3.3	looseMarshal	591
6.92.3.4	looseUnmarshal	592
6.92.3.5	tightMarshal1	592
6.92.3.6	tightMarshal2	592
6.92.3.7	tightUnmarshal	593
6.93	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	
	Class Reference	594
6.93.1	Detailed Description	594
6.93.2	Constructor & Destructor Documentation	595
6.93.2.1	ActiveMQTextMessageMarshaller	595
6.93.2.2	~ActiveMQTextMessageMarshaller	595
6.93.3	Member Function Documentation	595
6.93.3.1	createObject	595
6.93.3.2	getDataStructureType	595
6.93.3.3	looseMarshal	595
6.93.3.4	looseUnmarshal	596
6.93.3.5	tightMarshal1	596
6.93.3.6	tightMarshal2	596
6.93.3.7	tightUnmarshal	597
6.94	activemq::commands::ActiveMQTopic Class Reference	598
6.94.1	Constructor & Destructor Documentation	599
6.94.1.1	ActiveMQTopic	599
6.94.1.2	ActiveMQTopic	599
6.94.1.3	~ActiveMQTopic	599
6.94.2	Member Function Documentation	599
6.94.2.1	clone	599
6.94.2.2	cloneDataStructure	599
6.94.2.3	copy	599
6.94.2.4	copyDataStructure	599
6.94.2.5	equals	600
6.94.2.6	getCMSDestination	600
6.94.2.7	getCMSProperties	600
6.94.2.8	getDataStructureType	600
6.94.2.9	getDestinationType	601
6.94.2.10	getTopicName	601

6.94.2.11	toString	601
6.94.3	Field Documentation	601
6.94.3.1	ID_ACTIVEMQTOPIC	601
6.95	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference	602
6.95.1	Detailed Description	602
6.95.2	Constructor & Destructor Documentation	603
6.95.2.1	ActiveMQTopicMarshaller	603
6.95.2.2	~ActiveMQTopicMarshaller	603
6.95.3	Member Function Documentation	603
6.95.3.1	createObject	603
6.95.3.2	getDataStructureType	603
6.95.3.3	looseMarshal	603
6.95.3.4	looseUnmarshal	604
6.95.3.5	tightMarshal1	604
6.95.3.6	tightMarshal2	604
6.95.3.7	tightUnmarshal	605
6.96	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference	606
6.96.1	Detailed Description	606
6.96.2	Constructor & Destructor Documentation	607
6.96.2.1	ActiveMQTopicMarshaller	607
6.96.2.2	~ActiveMQTopicMarshaller	607
6.96.3	Member Function Documentation	607
6.96.3.1	createObject	607
6.96.3.2	getDataStructureType	607
6.96.3.3	looseMarshal	607
6.96.3.4	looseUnmarshal	608
6.96.3.5	tightMarshal1	608
6.96.3.6	tightMarshal2	608
6.96.3.7	tightUnmarshal	609
6.97	activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference	610
6.97.1	Detailed Description	610
6.97.2	Constructor & Destructor Documentation	611
6.97.2.1	ActiveMQTopicMarshaller	611
6.97.2.2	~ActiveMQTopicMarshaller	611

6.97.3	Member Function Documentation	611
6.97.3.1	createObject	611
6.97.3.2	getDataStructureType	611
6.97.3.3	looseMarshal	611
6.97.3.4	looseUnmarshal	612
6.97.3.5	tightMarshal1	612
6.97.3.6	tightMarshal2	612
6.97.3.7	tightUnmarshal	613
6.98	activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference	614
6.98.1	Detailed Description	614
6.98.2	Constructor & Destructor Documentation	615
6.98.2.1	ActiveMQTopicMarshaller	615
6.98.2.2	~ActiveMQTopicMarshaller	615
6.98.3	Member Function Documentation	615
6.98.3.1	createObject	615
6.98.3.2	getDataStructureType	615
6.98.3.3	looseMarshal	615
6.98.3.4	looseUnmarshal	616
6.98.3.5	tightMarshal1	616
6.98.3.6	tightMarshal2	616
6.98.3.7	tightUnmarshal	617
6.99	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference	618
6.99.1	Detailed Description	618
6.99.2	Constructor & Destructor Documentation	619
6.99.2.1	ActiveMQTopicMarshaller	619
6.99.2.2	~ActiveMQTopicMarshaller	619
6.99.3	Member Function Documentation	619
6.99.3.1	createObject	619
6.99.3.2	getDataStructureType	619
6.99.3.3	looseMarshal	619
6.99.3.4	looseUnmarshal	620
6.99.3.5	tightMarshal1	620
6.99.3.6	tightMarshal2	620
6.99.3.7	tightUnmarshal	621
6.100	activemq::core::ActiveMQTransactionContext Class Reference	622

6.100.1 Detailed Description	622
6.100.2 Constructor & Destructor Documentation	623
6.100.2.1 ActiveMQTransactionContext	623
6.100.2.2 ~ActiveMQTransactionContext	623
6.100.3 Member Function Documentation	623
6.100.3.1 addSynchronization	623
6.100.3.2 begin	623
6.100.3.3 commit	623
6.100.3.4 getMaximumRedeliveries	624
6.100.3.5 getRedeliveryDelay	624
6.100.3.6 getTransactionId	624
6.100.3.7 isInTransaction	624
6.100.3.8 removeSynchronization	624
6.100.3.9 rollback	625
6.101decaf::lang::Appendable Class Reference	626
6.101.1 Constructor & Destructor Documentation	626
6.101.1.1 ~Appendable	626
6.101.2 Member Function Documentation	626
6.101.2.1 append	626
6.101.2.2 append	627
6.101.2.3 append	627
6.102decaf::internal::AprPool Class Reference	628
6.102.1 Detailed Description	628
6.102.2 Constructor & Destructor Documentation	628
6.102.2.1 AprPool	628
6.102.2.2 ~AprPool	628
6.102.3 Member Function Documentation	628
6.102.3.1 cleanup	628
6.102.3.2 getAprPool	628
6.102.3.3 getGlobalPool	629
6.103decaf::util::concurrent::atomic::AtomicBoolean Class Reference	630
6.103.1 Detailed Description	630
6.103.2 Constructor & Destructor Documentation	630
6.103.2.1 AtomicBoolean	630
6.103.2.2 AtomicBoolean	630
6.103.2.3 ~AtomicBoolean	631

6.103.3 Member Function Documentation	631
6.103.3.1 compareAndSet	631
6.103.3.2 get	631
6.103.3.3 getAndSet	631
6.103.3.4 set	631
6.103.3.5 toString	632
6.104 decaf::util::concurrent::atomic::AtomicInteger Class Reference	633
6.104.1 Detailed Description	634
6.104.2 Constructor & Destructor Documentation	634
6.104.2.1 AtomicInteger	634
6.104.2.2 AtomicInteger	634
6.104.2.3 ~AtomicInteger	634
6.104.3 Member Function Documentation	634
6.104.3.1 addAndGet	634
6.104.3.2 compareAndSet	635
6.104.3.3 decrementAndGet	635
6.104.3.4 doubleValue	635
6.104.3.5 floatValue	635
6.104.3.6 get	636
6.104.3.7 getAndAdd	636
6.104.3.8 getAndDecrement	636
6.104.3.9 getAndIncrement	636
6.104.3.10 getAndSet	636
6.104.3.11 incrementAndGet	637
6.104.3.12 intValue	637
6.104.3.13 longValue	637
6.104.3.14 set	637
6.104.3.15 toString	637
6.105 decaf::lang::AtomicRefCounter Class Reference	638
6.105.1 Constructor & Destructor Documentation	639
6.105.1.1 AtomicRefCounter	639
6.105.1.2 AtomicRefCounter	639
6.105.2 Member Function Documentation	639
6.105.2.1 release	639
6.105.2.2 swap	640
6.106 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference	641

6.106.1 Detailed Description	641
6.106.2 Constructor & Destructor Documentation	642
6.106.2.1 AtomicReference	642
6.106.2.2 AtomicReference	642
6.106.2.3 ~AtomicReference	642
6.106.3 Member Function Documentation	642
6.106.3.1 compareAndSet	642
6.106.3.2 get	642
6.106.3.3 getAndSet	642
6.106.3.4 set	643
6.106.3.5 toString	643
6.107activemq::transport::failover::BackupTransport Class Reference	644
6.107.1 Constructor & Destructor Documentation	644
6.107.1.1 BackupTransport	644
6.107.1.2 ~BackupTransport	644
6.107.2 Member Function Documentation	644
6.107.2.1 getTransport	644
6.107.2.2 getUri	645
6.107.2.3 isClosed	645
6.107.2.4 onException	645
6.107.2.5 setClosed	645
6.107.2.6 setTransport	645
6.107.2.7 setUri	646
6.108activemq::transport::failover::BackupTransportPool Class Reference	647
6.108.1 Constructor & Destructor Documentation	648
6.108.1.1 BackupTransportPool	648
6.108.1.2 BackupTransportPool	648
6.108.1.3 ~BackupTransportPool	648
6.108.2 Member Function Documentation	648
6.108.2.1 getBackup	648
6.108.2.2 getBackupPoolSize	648
6.108.2.3 isEnabled	648
6.108.2.4 isPending	649
6.108.2.5 iterate	649
6.108.2.6 setBackupPoolSize	649
6.108.2.7 setEnabled	649

6.108.3 Friends And Related Function Documentation	649
6.108.3.1 BackupTransport	649
6.109 activemq::commands::BaseCommand Class Reference	650
6.109.1 Constructor & Destructor Documentation	651
6.109.1.1 BaseCommand	651
6.109.1.2 ~BaseCommand	651
6.109.2 Member Function Documentation	651
6.109.2.1 copyDataStructure	651
6.109.2.2 equals	651
6.109.2.3 getCommandId	652
6.109.2.4 isBrokerInfo	653
6.109.2.5 isConnectionInfo	653
6.109.2.6 isConsumerInfo	653
6.109.2.7 isKeepAliveInfo	653
6.109.2.8 isMessage	653
6.109.2.9 isMessageAck	653
6.109.2.10 isMessageDispatch	653
6.109.2.11 isMessageDispatchNotification	654
6.109.2.12 sProducerAck	654
6.109.2.13 sProducerInfo	654
6.109.2.14 sRemoveInfo	654
6.109.2.15 sRemoveSubscriptionInfo	654
6.109.2.16 sResponse	654
6.109.2.17 sResponseRequired	654
6.109.2.18 sShutdownInfo	655
6.109.2.19 sTransactionInfo	655
6.109.2.20 sWireFormatInfo	655
6.109.2.21 setCommandId	655
6.109.2.22 setResponseRequired	655
6.109.2.23 oString	655
6.110 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller Class Reference	657
6.110.1 Detailed Description	657
6.110.2 Constructor & Destructor Documentation	658
6.110.2.1 BaseCommandMarshaller	658
6.110.2.2 ~BaseCommandMarshaller	658
6.110.3 Member Function Documentation	658

6.110.3.1 looseMarshal	658
6.110.3.2 looseUnmarshal	659
6.110.3.3 tightMarshal1	660
6.110.3.4 tightMarshal2	661
6.110.3.5 tightUnmarshal	662
6.111activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller Class Reference	664
6.111.1 Detailed Description	664
6.111.2 Constructor & Destructor Documentation	665
6.111.2.1 BaseCommandMarshaller	665
6.111.2.2 ~BaseCommandMarshaller	665
6.111.3 Member Function Documentation	665
6.111.3.1 looseMarshal	665
6.111.3.2 looseUnmarshal	666
6.111.3.3 tightMarshal1	667
6.111.3.4 tightMarshal2	668
6.111.3.5 tightUnmarshal	669
6.112activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller Class Reference	671
6.112.1 Detailed Description	671
6.112.2 Constructor & Destructor Documentation	672
6.112.2.1 BaseCommandMarshaller	672
6.112.2.2 ~BaseCommandMarshaller	672
6.112.3 Member Function Documentation	672
6.112.3.1 looseMarshal	672
6.112.3.2 looseUnmarshal	673
6.112.3.3 tightMarshal1	674
6.112.3.4 tightMarshal2	675
6.112.3.5 tightUnmarshal	676
6.113activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference	678
6.113.1 Detailed Description	678
6.113.2 Constructor & Destructor Documentation	679
6.113.2.1 BaseCommandMarshaller	679
6.113.2.2 ~BaseCommandMarshaller	679
6.113.3 Member Function Documentation	679
6.113.3.1 looseMarshal	679

6.113.3.2 looseUnmarshal	680
6.113.3.3 tightMarshal1	681
6.113.3.4 tightMarshal2	682
6.113.3.5 tightUnmarshal	683
6.114activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference	685
6.114.1 Detailed Description	685
6.114.2 Constructor & Destructor Documentation	686
6.114.2.1 BaseCommandMarshaller	686
6.114.2.2 ~BaseCommandMarshaller	686
6.114.3 Member Function Documentation	686
6.114.3.1 looseMarshal	686
6.114.3.2 looseUnmarshal	687
6.114.3.3 tightMarshal1	688
6.114.3.4 tightMarshal2	689
6.114.3.5 tightUnmarshal	690
6.115activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference	692
6.115.1 Detailed Description	698
6.115.2 Constructor & Destructor Documentation	698
6.115.2.1 ~BaseDataStreamMarshaller	698
6.115.3 Member Function Documentation	698
6.115.3.1 looseMarshal	698
6.115.3.2 looseMarshalBrokerError	698
6.115.3.3 looseMarshalCachedObject	699
6.115.3.4 looseMarshalLong	699
6.115.3.5 looseMarshalNestedObject	699
6.115.3.6 looseMarshalObjectArray	700
6.115.3.7 looseMarshalString	700
6.115.3.8 looseUnmarshal	701
6.115.3.9 looseUnmarshalBrokerError	701
6.115.3.10looseUnmarshalByteArray	701
6.115.3.11looseUnmarshalCachedObject	702
6.115.3.12looseUnmarshalConstByteArray	702
6.115.3.13looseUnmarshalLong	702
6.115.3.14looseUnmarshalNestedObject	703
6.115.3.15looseUnmarshalString	703

6.115.3.16	readAsciiString	703
6.115.3.17	rightMarshal1	704
6.115.3.18	rightMarshal2	704
6.115.3.19	rightMarshalBrokerError1	705
6.115.3.20	rightMarshalBrokerError2	705
6.115.3.21	rightMarshalCachedObject1	705
6.115.3.22	rightMarshalCachedObject2	706
6.115.3.23	rightMarshalLong1	706
6.115.3.24	rightMarshalLong2	707
6.115.3.25	rightMarshalNestedObject1	707
6.115.3.26	rightMarshalNestedObject2	707
6.115.3.27	rightMarshalObjectArray1	708
6.115.3.28	rightMarshalObjectArray2	708
6.115.3.29	rightMarshalString1	709
6.115.3.30	rightMarshalString2	709
6.115.3.31	rightUnmarshal	709
6.115.3.32	rightUnmarshalBrokerError	710
6.115.3.33	rightUnmarshalByteArray	710
6.115.3.34	rightUnmarshalCachedObject	711
6.115.3.35	rightUnmarshalConstByteArray	711
6.115.3.36	rightUnmarshalLong	711
6.115.3.37	rightUnmarshalNestedObject	712
6.115.3.38	rightUnmarshalString	712
6.115.3.39	toHexFromBytes	713
6.115.3.40	toString	713
6.115.3.41	toString	713
6.115.3.42	toString	713
6.116	activemq::commands::BaseDataStructure Class Reference	715
6.116.1	Constructor & Destructor Documentation	716
6.116.1.1	~BaseDataStructure	716
6.116.2	Member Function Documentation	716
6.116.2.1	afterMarshal	716
6.116.2.2	afterUnmarshal	716
6.116.2.3	beforeMarshal	716
6.116.2.4	beforeUnmarshal	716
6.116.2.5	copyDataStructure	717

6.116.2.6 equals	717
6.116.2.7 getMarshaledForm	717
6.116.2.8 isMarshalAware	717
6.116.2.9 setMarshaledForm	718
6.116.2.10 toString	718
6.117binary_function Class Reference	720
6.118decaf::net::BindException Class Reference	721
6.118.1 Constructor & Destructor Documentation	721
6.118.1.1 BindException	721
6.118.1.2 BindException	721
6.118.1.3 BindException	722
6.118.1.4 BindException	722
6.118.1.5 BindException	722
6.118.1.6 BindException	722
6.118.1.7 ~BindException	723
6.118.2 Member Function Documentation	723
6.118.2.1 clone	723
6.119decaf::io::BlockingByteArrayInputStream Class Reference	724
6.119.1 Detailed Description	725
6.119.2 Constructor & Destructor Documentation	725
6.119.2.1 BlockingByteArrayInputStream	725
6.119.2.2 BlockingByteArrayInputStream	726
6.119.2.3 ~BlockingByteArrayInputStream	726
6.119.3 Member Function Documentation	726
6.119.3.1 available	726
6.119.3.2 close	726
6.119.3.3 lock	726
6.119.3.4 mark	727
6.119.3.5 markSupported	727
6.119.3.6 notify	727
6.119.3.7 notifyAll	727
6.119.3.8 read	728
6.119.3.9 read	728
6.119.3.10 reset	728
6.119.3.11 setByteArray	729
6.119.3.12 skip	729

6.119.3.13	tryLock	730
6.119.3.14	unlock	730
6.119.3.15	wait	730
6.119.3.16	wait	731
6.119.3.17	wait	731
6.120	decaf::lang::Boolean Class Reference	732
6.120.1	Constructor & Destructor Documentation	733
6.120.1.1	Boolean	733
6.120.1.2	Boolean	733
6.120.1.3	~Boolean	733
6.120.2	Member Function Documentation	733
6.120.2.1	booleanValue	733
6.120.2.2	compareTo	733
6.120.2.3	compareTo	734
6.120.2.4	equals	734
6.120.2.5	equals	734
6.120.2.6	operator<	734
6.120.2.7	operator<	735
6.120.2.8	operator==	735
6.120.2.9	operator==	735
6.120.2.10	parseBoolean	735
6.120.2.11	toString	736
6.120.2.12	toString	736
6.120.2.13	valueOf	736
6.120.2.14	valueOf	736
6.120.3	Field Documentation	736
6.120.3.1	_FALSE	736
6.120.3.2	_TRUE	736
6.121	activemq::commands::BooleanExpression Class Reference	737
6.121.1	Constructor & Destructor Documentation	737
6.121.1.1	BooleanExpression	737
6.121.1.2	~BooleanExpression	737
6.121.2	Member Function Documentation	737
6.121.2.1	cloneDataStructure	737
6.121.2.2	copyDataStructure	738
6.121.2.3	equals	738

6.121.2.4 toString	738
6.122activemq::wireformat::openwire::utils::BooleanStream Class Reference	739
6.122.1 Detailed Description	739
6.122.2 Constructor & Destructor Documentation	740
6.122.2.1 BooleanStream	740
6.122.2.2 ~BooleanStream	740
6.122.3 Member Function Documentation	740
6.122.3.1 clear	740
6.122.3.2 marshal	740
6.122.3.3 marshal	740
6.122.3.4 marshalledSize	740
6.122.3.5 readBoolean	740
6.122.3.6 unmarshal	741
6.122.3.7 writeBoolean	741
6.123decaf::util::concurrent::BrokenBarrierException Class Reference	742
6.123.1 Constructor & Destructor Documentation	742
6.123.1.1 BrokenBarrierException	742
6.123.1.2 BrokenBarrierException	742
6.123.1.3 BrokenBarrierException	743
6.123.1.4 BrokenBarrierException	743
6.123.1.5 BrokenBarrierException	743
6.123.1.6 BrokenBarrierException	743
6.123.1.7 ~BrokenBarrierException	744
6.123.2 Member Function Documentation	744
6.123.2.1 clone	744
6.124activemq::commands::BrokerError Class Reference	745
6.124.1 Detailed Description	746
6.124.2 Constructor & Destructor Documentation	746
6.124.2.1 BrokerError	746
6.124.2.2 ~BrokerError	746
6.124.3 Member Function Documentation	746
6.124.3.1 cloneDataStructure	746
6.124.3.2 copyDataStructure	746
6.124.3.3 getCause	747
6.124.3.4 getDataStructureType	747
6.124.3.5 getExceptionClass	747

6.124.3.6	getMessage	747
6.124.3.7	getStackTraceElements	747
6.124.3.8	setCause	748
6.124.3.9	setExceptionClass	748
6.124.3.10	setMessage	748
6.124.3.11	setStackTraceElements	748
6.124.3.12	visit	748
6.125	activemq::exceptions::BrokerException Class Reference	749
6.125.1	Constructor & Destructor Documentation	749
6.125.1.1	BrokerException	749
6.125.1.2	BrokerException	749
6.125.1.3	BrokerException	749
6.125.1.4	BrokerException	749
6.125.1.5	BrokerException	749
6.125.1.6	~BrokerException	749
6.125.2	Member Function Documentation	749
6.125.2.1	clone	749
6.126	activemq::commands::BrokerId Class Reference	751
6.126.1	Member Typedef Documentation	752
6.126.1.1	COMPARATOR	752
6.126.2	Constructor & Destructor Documentation	752
6.126.2.1	BrokerId	752
6.126.2.2	BrokerId	752
6.126.2.3	~BrokerId	752
6.126.3	Member Function Documentation	752
6.126.3.1	cloneDataStructure	752
6.126.3.2	compareTo	752
6.126.3.3	copyDataStructure	752
6.126.3.4	equals	752
6.126.3.5	equals	752
6.126.3.6	getDataStructureType	753
6.126.3.7	getValue	753
6.126.3.8	getValue	753
6.126.3.9	operator<	753
6.126.3.10	operator=	753
6.126.3.11	operator==	753

6.126.3.12 set Value	753
6.126.3.13 toString	753
6.126.4 Field Documentation	753
6.126.4.1 ID_BROKERID	753
6.126.4.2 value	753
6.127activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference	755
6.127.1 Detailed Description	755
6.127.2 Constructor & Destructor Documentation	756
6.127.2.1 BrokerIdMarshaller	756
6.127.2.2 ~BrokerIdMarshaller	756
6.127.3 Member Function Documentation	756
6.127.3.1 createObject	756
6.127.3.2 getDataStructureType	756
6.127.3.3 looseMarshal	756
6.127.3.4 looseUnmarshal	757
6.127.3.5 tightMarshal1	757
6.127.3.6 tightMarshal2	757
6.127.3.7 tightUnmarshal	758
6.128activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference	759
6.128.1 Detailed Description	759
6.128.2 Constructor & Destructor Documentation	760
6.128.2.1 BrokerIdMarshaller	760
6.128.2.2 ~BrokerIdMarshaller	760
6.128.3 Member Function Documentation	760
6.128.3.1 createObject	760
6.128.3.2 getDataStructureType	760
6.128.3.3 looseMarshal	760
6.128.3.4 looseUnmarshal	761
6.128.3.5 tightMarshal1	761
6.128.3.6 tightMarshal2	761
6.128.3.7 tightUnmarshal	762
6.129activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference	763
6.129.1 Detailed Description	763
6.129.2 Constructor & Destructor Documentation	764
6.129.2.1 BrokerIdMarshaller	764
6.129.2.2 ~BrokerIdMarshaller	764

6.129.3 Member Function Documentation	764
6.129.3.1 createObject	764
6.129.3.2 getDataStructureType	764
6.129.3.3 looseMarshal	764
6.129.3.4 looseUnmarshal	765
6.129.3.5 tightMarshal1	765
6.129.3.6 tightMarshal2	765
6.129.3.7 tightUnmarshal	766
6.130activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference	767
6.130.1 Detailed Description	767
6.130.2 Constructor & Destructor Documentation	768
6.130.2.1 BrokerIdMarshaller	768
6.130.2.2 ~BrokerIdMarshaller	768
6.130.3 Member Function Documentation	768
6.130.3.1 createObject	768
6.130.3.2 getDataStructureType	768
6.130.3.3 looseMarshal	768
6.130.3.4 looseUnmarshal	769
6.130.3.5 tightMarshal1	769
6.130.3.6 tightMarshal2	769
6.130.3.7 tightUnmarshal	770
6.131activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference	771
6.131.1 Detailed Description	771
6.131.2 Constructor & Destructor Documentation	772
6.131.2.1 BrokerIdMarshaller	772
6.131.2.2 ~BrokerIdMarshaller	772
6.131.3 Member Function Documentation	772
6.131.3.1 createObject	772
6.131.3.2 getDataStructureType	772
6.131.3.3 looseMarshal	772
6.131.3.4 looseUnmarshal	773
6.131.3.5 tightMarshal1	773
6.131.3.6 tightMarshal2	773
6.131.3.7 tightUnmarshal	774
6.132activemq::commands::BrokerInfo Class Reference	775
6.132.1 Constructor & Destructor Documentation	777

6.132.1.1 BrokerInfo	777
6.132.1.2 BrokerInfo	777
6.132.1.3 ~BrokerInfo	777
6.132.2 Member Function Documentation	777
6.132.2.1 cloneDataStructure	777
6.132.2.2 copyDataStructure	777
6.132.2.3 equals	777
6.132.2.4 getBrokerId	778
6.132.2.5 getBrokerId	778
6.132.2.6 getBrokerName	778
6.132.2.7 getBrokerName	778
6.132.2.8 getBrokerUploadUrl	778
6.132.2.9 getBrokerUploadUrl	778
6.132.2.10 getBrokerURL	778
6.132.2.11 getBrokerURL	778
6.132.2.12 getConnectionId	778
6.132.2.13 getDataStructureType	778
6.132.2.14 getNetworkProperties	779
6.132.2.15 getNetworkProperties	779
6.132.2.16 getPeerBrokerInfos	779
6.132.2.17 getPeerBrokerInfos	779
6.132.2.18 sBrokerInfo	779
6.132.2.19 sDuplexConnection	780
6.132.2.20 sFaultTolerantConfiguration	780
6.132.2.21 sIsMasterBroker	780
6.132.2.22 sNetworkConnection	780
6.132.2.23 sSlaveBroker	780
6.132.2.24 operator=	780
6.132.2.25 setBrokerId	780
6.132.2.26 setBrokerName	780
6.132.2.27 setBrokerUploadUrl	780
6.132.2.28 setBrokerURL	780
6.132.2.29 setConnectionId	780
6.132.2.30 setDuplexConnection	780
6.132.2.31 setFaultTolerantConfiguration	780
6.132.2.32 setMasterBroker	780

6.132.2.33	setNetworkConnection	780
6.132.2.34	setNetworkProperties	780
6.132.2.35	setPeerBrokerInfos	780
6.132.2.36	setSlaveBroker	780
6.132.2.37	toString	780
6.132.2.38	visit	781
6.132.3	Field Documentation	782
6.132.3.1	brokerId	782
6.132.3.2	brokerName	782
6.132.3.3	brokerUploadUrl	782
6.132.3.4	brokerURL	782
6.132.3.5	connectionId	782
6.132.3.6	duplexConnection	782
6.132.3.7	faultTolerantConfiguration	782
6.132.3.8	ID_BROKERINFO	782
6.132.3.9	masterBroker	782
6.132.3.10	networkConnection	782
6.132.3.11	networkProperties	782
6.132.3.12	peerBrokerInfos	782
6.132.3.13	slaveBroker	782
6.133	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference	783
6.133.1	Detailed Description	783
6.133.2	Constructor & Destructor Documentation	784
6.133.2.1	BrokerInfoMarshaller	784
6.133.2.2	~BrokerInfoMarshaller	784
6.133.3	Member Function Documentation	784
6.133.3.1	createObject	784
6.133.3.2	getDataStructureType	784
6.133.3.3	looseMarshal	784
6.133.3.4	looseUnmarshal	785
6.133.3.5	tightMarshal1	785
6.133.3.6	tightMarshal2	785
6.133.3.7	tightUnmarshal	786
6.134	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference	787
6.134.1	Detailed Description	787
6.134.2	Constructor & Destructor Documentation	788

6.134.2.1 BrokerInfoMarshaller	788
6.134.2.2 ~BrokerInfoMarshaller	788
6.134.3 Member Function Documentation	788
6.134.3.1 createObject	788
6.134.3.2 getDataStructureType	788
6.134.3.3 looseMarshal	788
6.134.3.4 looseUnmarshal	789
6.134.3.5 tightMarshal1	789
6.134.3.6 tightMarshal2	789
6.134.3.7 tightUnmarshal	790
6.135activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference	791
6.135.1 Detailed Description	791
6.135.2 Constructor & Destructor Documentation	792
6.135.2.1 BrokerInfoMarshaller	792
6.135.2.2 ~BrokerInfoMarshaller	792
6.135.3 Member Function Documentation	792
6.135.3.1 createObject	792
6.135.3.2 getDataStructureType	792
6.135.3.3 looseMarshal	792
6.135.3.4 looseUnmarshal	793
6.135.3.5 tightMarshal1	793
6.135.3.6 tightMarshal2	793
6.135.3.7 tightUnmarshal	794
6.136activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference	795
6.136.1 Detailed Description	795
6.136.2 Constructor & Destructor Documentation	796
6.136.2.1 BrokerInfoMarshaller	796
6.136.2.2 ~BrokerInfoMarshaller	796
6.136.3 Member Function Documentation	796
6.136.3.1 createObject	796
6.136.3.2 getDataStructureType	796
6.136.3.3 looseMarshal	796
6.136.3.4 looseUnmarshal	797
6.136.3.5 tightMarshal1	797
6.136.3.6 tightMarshal2	797
6.136.3.7 tightUnmarshal	798

6.137	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference	799
6.137.1	Detailed Description	799
6.137.2	Constructor & Destructor Documentation	800
6.137.2.1	BrokerInfoMarshaller	800
6.137.2.2	~BrokerInfoMarshaller	800
6.137.3	Member Function Documentation	800
6.137.3.1	createObject	800
6.137.3.2	getDataStructureType	800
6.137.3.3	looseMarshal	800
6.137.3.4	looseUnmarshal	801
6.137.3.5	tightMarshal1	801
6.137.3.6	tightMarshal2	801
6.137.3.7	tightUnmarshal	802
6.138	decaf::nio::Buffer Class Reference	803
6.138.1	Detailed Description	804
6.138.2	Constructor & Destructor Documentation	805
6.138.2.1	Buffer	805
6.138.2.2	Buffer	805
6.138.2.3	~Buffer	805
6.138.3	Member Function Documentation	805
6.138.3.1	capacity	805
6.138.3.2	clear	805
6.138.3.3	flip	806
6.138.3.4	hasRemaining	806
6.138.3.5	isReadOnly	806
6.138.3.6	limit	806
6.138.3.7	limit	807
6.138.3.8	mark	807
6.138.3.9	position	807
6.138.3.10	position	807
6.138.3.11	remaining	808
6.138.3.12	reset	808
6.138.3.13	rewind	808
6.138.4	Field Documentation	808
6.138.4.1	_capacity	808
6.138.4.2	_limit	808

6.138.4.3	_mark	808
6.138.4.4	_markSet	808
6.138.4.5	_position	808
6.139	decaf::io::BufferedInputStream Class Reference	809
6.139.1	Detailed Description	810
6.139.2	Constructor & Destructor Documentation	810
6.139.2.1	BufferedInputStream	810
6.139.2.2	BufferedInputStream	810
6.139.2.3	~BufferedInputStream	810
6.139.3	Member Function Documentation	810
6.139.3.1	available	810
6.139.3.2	close	811
6.139.3.3	mark	811
6.139.3.4	markSupported	811
6.139.3.5	read	811
6.139.3.6	read	812
6.139.3.7	reset	812
6.139.3.8	skip	812
6.140	decaf::io::BufferedOutputStream Class Reference	814
6.140.1	Detailed Description	814
6.140.2	Constructor & Destructor Documentation	814
6.140.2.1	BufferedOutputStream	814
6.140.2.2	BufferedOutputStream	815
6.140.2.3	~BufferedOutputStream	815
6.140.3	Member Function Documentation	815
6.140.3.1	close	815
6.140.3.2	flush	815
6.140.3.3	write	815
6.140.3.4	write	816
6.140.3.5	write	816
6.141	decaf::net::BufferedSocket Class Reference	817
6.141.1	Detailed Description	818
6.141.2	Constructor & Destructor Documentation	818
6.141.2.1	BufferedSocket	818
6.141.2.2	~BufferedSocket	818
6.141.3	Member Function Documentation	818

6.141.3.1 close	818
6.141.3.2 connect	819
6.141.3.3 getInputStream	819
6.141.3.4 getKeepAlive	819
6.141.3.5 getOutputStream	819
6.141.3.6 getReceiveBufferSize	820
6.141.3.7 getReuseAddress	820
6.141.3.8 getSendBufferSize	820
6.141.3.9 getSoLinger	820
6.141.3.10getSoTimeout	821
6.141.3.11isConnected	821
6.141.3.12setKeepAlive	821
6.141.3.13setReceiveBufferSize	822
6.141.3.14setReuseAddress	822
6.141.3.15setSendBufferSize	822
6.141.3.16setSoLinger	822
6.141.3.17setSoTimeout	823
6.142decaf::internal::nio::BufferFactory Class Reference	824
6.142.1 Detailed Description	825
6.142.2 Constructor & Destructor Documentation	826
6.142.2.1 ~BufferFactory	826
6.142.3 Member Function Documentation	826
6.142.3.1 createByteBuffer	826
6.142.3.2 createByteBuffer	826
6.142.3.3 createByteBuffer	826
6.142.3.4 createCharBuffer	827
6.142.3.5 createCharBuffer	827
6.142.3.6 createCharBuffer	827
6.142.3.7 createDoubleBuffer	828
6.142.3.8 createDoubleBuffer	828
6.142.3.9 createDoubleBuffer	828
6.142.3.10createFloatBuffer	829
6.142.3.11createFloatBuffer	829
6.142.3.12reateFloatBuffer	829
6.142.3.13reateIntBuffer	830
6.142.3.14reateIntBuffer	830

6.142.3.15	createIntBuffer	830
6.142.3.16	createLongBuffer	831
6.142.3.17	createLongBuffer	831
6.142.3.18	createLongBuffer	831
6.142.3.19	createShortBuffer	832
6.142.3.20	createShortBuffer	832
6.142.3.21	createShortBuffer	832
6.143	decaf::nio::BufferOverflowException Class Reference	834
6.143.1	Constructor & Destructor Documentation	834
6.143.1.1	BufferOverflowException	834
6.143.1.2	BufferOverflowException	834
6.143.1.3	BufferOverflowException	835
6.143.1.4	BufferOverflowException	835
6.143.1.5	BufferOverflowException	835
6.143.1.6	BufferOverflowException	835
6.143.1.7	~BufferOverflowException	836
6.143.2	Member Function Documentation	836
6.143.2.1	clone	836
6.144	decaf::nio::BufferUnderflowException Class Reference	837
6.144.1	Constructor & Destructor Documentation	837
6.144.1.1	BufferUnderflowException	837
6.144.1.2	BufferUnderflowException	837
6.144.1.3	BufferUnderflowException	838
6.144.1.4	BufferUnderflowException	838
6.144.1.5	BufferUnderflowException	838
6.144.1.6	BufferUnderflowException	838
6.144.1.7	~BufferUnderflowException	839
6.144.2	Member Function Documentation	839
6.144.2.1	clone	839
6.145	decaf::lang::Byte Class Reference	840
6.145.1	Constructor & Destructor Documentation	841
6.145.1.1	Byte	841
6.145.1.2	Byte	842
6.145.1.3	~Byte	842
6.145.2	Member Function Documentation	842
6.145.2.1	byteValue	842

6.145.2.2 compareTo	842
6.145.2.3 compareTo	842
6.145.2.4 decode	843
6.145.2.5 doubleValue	843
6.145.2.6 equals	843
6.145.2.7 equals	844
6.145.2.8 floatValue	844
6.145.2.9 intValue	844
6.145.2.10 longValue	844
6.145.2.11 operator<	844
6.145.2.12 operator<	845
6.145.2.13 operator==	845
6.145.2.14 operator==	845
6.145.2.15 parseByte	845
6.145.2.16 parseByte	846
6.145.2.17 shortValue	846
6.145.2.18 toString	846
6.145.2.19 toString	847
6.145.2.20 valueOf	847
6.145.2.21 valueOf	847
6.145.2.22 valueOf	847
6.145.3 Field Documentation	848
6.145.3.1 MAX_VALUE	848
6.145.3.2 MIN_VALUE	848
6.145.3.3 SIZE	848
6.146 decaf::internal::util::ByteArrayAdapter Class Reference	849
6.146.1 Detailed Description	853
6.146.2 Constructor & Destructor Documentation	853
6.146.2.1 ByteArrayAdapter	853
6.146.2.2 ByteArrayAdapter	853
6.146.2.3 ByteArrayAdapter	854
6.146.2.4 ByteArrayAdapter	854
6.146.2.5 ByteArrayAdapter	854
6.146.2.6 ByteArrayAdapter	855
6.146.2.7 ByteArrayAdapter	855
6.146.2.8 ByteArrayAdapter	855

6.146.2.9 ~ByteArrayAdapter	856
6.146.3 Member Function Documentation	856
6.146.3.1 clear	856
6.146.3.2 get	856
6.146.3.3 getByteArray	856
6.146.3.4 getCapacity	856
6.146.3.5 getChar	856
6.146.3.6 getCharArray	857
6.146.3.7 getCharCapacity	857
6.146.3.8 getDouble	857
6.146.3.9 getDoubleArray	858
6.146.3.10 getDoubleAt	858
6.146.3.11 getDoubleCapacity	858
6.146.3.12 getFloat	858
6.146.3.13 getFloatArray	859
6.146.3.14 getFloatAt	859
6.146.3.15 getFloatCapacity	859
6.146.3.16 getInt	860
6.146.3.17 getIntArray	860
6.146.3.18 getIntAt	860
6.146.3.19 getIntCapacity	861
6.146.3.20 getLong	861
6.146.3.21 getLongArray	861
6.146.3.22 getLongAt	861
6.146.3.23 getLongCapacity	862
6.146.3.24 getShort	862
6.146.3.25 getShortArray	862
6.146.3.26 getShortAt	863
6.146.3.27 getShortCapacity	863
6.146.3.28 operator[]	863
6.146.3.29 operator[]	863
6.146.3.30 put	864
6.146.3.31 putChar	864
6.146.3.32 putDouble	864
6.146.3.33 putDoubleAt	865
6.146.3.34 putFloat	865

6.146.3.35	putFloatAt	866
6.146.3.36	putInt	866
6.146.3.37	putIntAt	866
6.146.3.38	putLong	867
6.146.3.39	putLongAt	867
6.146.3.40	putShort	868
6.146.3.41	putShortAt	868
6.146.3.42	read	868
6.146.3.43	resize	869
6.146.3.44	write	869
6.147	decaf::internal::nio::ByteBuffer Class Reference	870
6.147.1	Detailed Description	873
6.147.2	Constructor & Destructor Documentation	874
6.147.2.1	ByteBuffer	874
6.147.2.2	ByteBuffer	875
6.147.2.3	ByteBuffer	875
6.147.2.4	ByteBuffer	875
6.147.2.5	~ByteBuffer	876
6.147.3	Member Function Documentation	876
6.147.3.1	array	876
6.147.3.2	arrayOffset	876
6.147.3.3	asCharBuffer	876
6.147.3.4	asDoubleBuffer	877
6.147.3.5	asFloatBuffer	877
6.147.3.6	asIntBuffer	877
6.147.3.7	asLongBuffer	878
6.147.3.8	asReadOnlyBuffer	878
6.147.3.9	asShortBuffer	878
6.147.3.10	compact	879
6.147.3.11	duplicate	879
6.147.3.12	get	879
6.147.3.13	get	880
6.147.3.14	getChar	880
6.147.3.15	getChar	880
6.147.3.16	getDouble	881
6.147.3.17	getDouble	881

6.147.3.18	getFloat	881
6.147.3.19	getFloat	882
6.147.3.20	getInt	882
6.147.3.21	getInt	883
6.147.3.22	getLong	883
6.147.3.23	getLong	883
6.147.3.24	getShort	884
6.147.3.25	getShort	884
6.147.3.26	hasArray	884
6.147.3.27	isReadOnly	884
6.147.3.28	put	885
6.147.3.29	put	885
6.147.3.30	putChar	886
6.147.3.31	putChar	886
6.147.3.32	putDouble	886
6.147.3.33	putDouble	887
6.147.3.34	putFloat	887
6.147.3.35	putFloat	888
6.147.3.36	putInt	888
6.147.3.37	putInt	889
6.147.3.38	putLong	889
6.147.3.39	putLong	889
6.147.3.40	putShort	890
6.147.3.41	putShort	890
6.147.3.42	setReadOnly	891
6.147.3.43	slice	891
6.148	decaf::io::ByteArrayInputStream Class Reference	892
6.148.1	Detailed Description	893
6.148.2	Constructor & Destructor Documentation	894
6.148.2.1	ByteArrayInputStream	894
6.148.2.2	ByteArrayInputStream	894
6.148.2.3	ByteArrayInputStream	894
6.148.2.4	~ByteArrayInputStream	894
6.148.3	Member Function Documentation	894
6.148.3.1	available	894
6.148.3.2	close	894

6.148.3.3 lock	895
6.148.3.4 mark	895
6.148.3.5 markSupported	895
6.148.3.6 notify	895
6.148.3.7 notifyAll	896
6.148.3.8 read	896
6.148.3.9 read	896
6.148.3.10 reset	897
6.148.3.11 setBuffer	897
6.148.3.12 setByteArray	897
6.148.3.13 skip	897
6.148.3.14 tryLock	898
6.148.3.15 unlock	898
6.148.3.16 wait	898
6.148.3.17 wait	899
6.148.3.18 wait	899
6.149 decaf::io::ByteArrayOutputStream Class Reference	900
6.149.1 Constructor & Destructor Documentation	901
6.149.1.1 ByteArrayOutputStream	901
6.149.1.2 ByteArrayOutputStream	901
6.149.1.3 ~ByteArrayOutputStream	902
6.149.2 Member Function Documentation	902
6.149.2.1 close	902
6.149.2.2 flush	902
6.149.2.3 lock	902
6.149.2.4 notify	902
6.149.2.5 notifyAll	903
6.149.2.6 reset	903
6.149.2.7 setBuffer	903
6.149.2.8 size	903
6.149.2.9 toByteArray	904
6.149.2.10 toByteArrayRef	904
6.149.2.11 toString	904
6.149.2.12 tryLock	904
6.149.2.13 unlock	904
6.149.2.14 wait	905

6.149.2.15wait	905
6.149.2.16wait	906
6.149.2.17write	906
6.149.2.18write	906
6.149.2.19write	907
6.149.2.20writeTo	907
6.150decaf::internal::nio::ByteArrayPerspective Class Reference	908
6.150.1 Detailed Description	909
6.150.2 Constructor & Destructor Documentation	909
6.150.2.1 ByteArrayPerspective	909
6.150.2.2 ByteArrayPerspective	909
6.150.2.3 ByteArrayPerspective	909
6.150.2.4 ByteArrayPerspective	910
6.150.2.5 ByteArrayPerspective	910
6.150.2.6 ByteArrayPerspective	910
6.150.2.7 ByteArrayPerspective	911
6.150.2.8 ByteArrayPerspective	911
6.150.2.9 ~ByteArrayPerspective	911
6.150.3 Member Function Documentation	911
6.150.3.1 getReferences	911
6.150.3.2 returnRef	912
6.150.3.3 takeRef	912
6.151decaf::nio::ByteBuffer Class Reference	913
6.151.1 Detailed Description	917
6.151.2 Constructor & Destructor Documentation	918
6.151.2.1 ByteBuffer	918
6.151.2.2 ~ByteBuffer	918
6.151.3 Member Function Documentation	918
6.151.3.1 allocate	918
6.151.3.2 array	918
6.151.3.3 arrayOffset	919
6.151.3.4 asCharBuffer	919
6.151.3.5 asDoubleBuffer	919
6.151.3.6 asFloatBuffer	920
6.151.3.7 asIntBuffer	920
6.151.3.8 asLongBuffer	920

6.151.3.9 asReadOnlyBuffer	921
6.151.3.10 asShortBuffer	921
6.151.3.11 compact	921
6.151.3.12 compareTo	922
6.151.3.13 duplicate	922
6.151.3.14 equals	922
6.151.3.15 get	922
6.151.3.16 get	923
6.151.3.17 get	923
6.151.3.18 get	924
6.151.3.19 getChar	924
6.151.3.20 getChar	924
6.151.3.21 getDouble	925
6.151.3.22 getDouble	925
6.151.3.23 getFloat	925
6.151.3.24 getFloat	926
6.151.3.25 getInt	926
6.151.3.26 getInt	926
6.151.3.27 getLong	927
6.151.3.28 getLong	927
6.151.3.29 getShort	927
6.151.3.30 getShort	928
6.151.3.31 hasArray	928
6.151.3.32 isReadOnly	928
6.151.3.33 operator<	928
6.151.3.34 operator==	929
6.151.3.35 put	929
6.151.3.36 put	929
6.151.3.37 put	930
6.151.3.38 put	930
6.151.3.39 put	930
6.151.3.40 putChar	931
6.151.3.41 putChar	931
6.151.3.42 putDouble	932
6.151.3.43 putDouble	932
6.151.3.44 putFloat	933

6.151.3.45	putFloat	933
6.151.3.46	putInt	933
6.151.3.47	putInt	934
6.151.3.48	putLong	934
6.151.3.49	putLong	935
6.151.3.50	putShort	935
6.151.3.51	putShort	935
6.151.3.52	lice	936
6.151.3.53	oString	936
6.151.3.54	wrap	936
6.151.3.55	wrap	937
6.152	cms::BytesMessage Class Reference	938
6.152.1	Detailed Description	940
6.152.2	Constructor & Destructor Documentation	941
6.152.2.1	~BytesMessage	941
6.152.3	Member Function Documentation	941
6.152.3.1	clone	941
6.152.3.2	getBodyBytes	941
6.152.3.3	getBodyLength	942
6.152.3.4	readBoolean	942
6.152.3.5	readByte	942
6.152.3.6	readBytes	943
6.152.3.7	readBytes	943
6.152.3.8	readChar	944
6.152.3.9	readDouble	944
6.152.3.10	readFloat	945
6.152.3.11	readInt	945
6.152.3.12	readLong	945
6.152.3.13	readShort	946
6.152.3.14	readString	946
6.152.3.15	readUnsignedShort	946
6.152.3.16	readUTF	947
6.152.3.17	reset	947
6.152.3.18	setBodyBytes	947
6.152.3.19	writeBoolean	948
6.152.3.20	writeByte	948

6.152.3.21	writeBytes	948
6.152.3.22	writeBytes	949
6.152.3.23	writeChar	949
6.152.3.24	writeDouble	949
6.152.3.25	writeFloat	950
6.152.3.26	writeInt	950
6.152.3.27	writeLong	950
6.152.3.28	writeShort	951
6.152.3.29	writeString	951
6.152.3.30	writeUnsignedShort	951
6.152.3.31	writeUTF	952
6.153	activemq::cmsutil::CachedConsumer Class Reference	953
6.153.1	Detailed Description	953
6.153.2	Constructor & Destructor Documentation	954
6.153.2.1	CachedConsumer	954
6.153.2.2	~CachedConsumer	954
6.153.3	Member Function Documentation	954
6.153.3.1	close	954
6.153.3.2	getMessageListener	954
6.153.3.3	getMessageSelector	954
6.153.3.4	receive	954
6.153.3.5	receive	955
6.153.3.6	receiveNoWait	955
6.153.3.7	setMessageListener	955
6.154	activemq::cmsutil::CachedProducer Class Reference	957
6.154.1	Detailed Description	958
6.154.2	Constructor & Destructor Documentation	958
6.154.2.1	CachedProducer	958
6.154.2.2	~CachedProducer	958
6.154.3	Member Function Documentation	958
6.154.3.1	close	958
6.154.3.2	getDeliveryMode	958
6.154.3.3	getDisableMessageID	959
6.154.3.4	getDisableMessageTimeStamp	959
6.154.3.5	getPriority	959
6.154.3.6	getTimeToLive	959

6.154.3.7 send	960
6.154.3.8 send	960
6.154.3.9 send	961
6.154.3.10 send	961
6.154.3.11 setDeliveryMode	962
6.154.3.12 setDisableMessageID	962
6.154.3.13 setDisableMessageTimeStamp	962
6.154.3.14 setPriority	962
6.154.3.15 setTimeToLive	963
6.155 decaf::util::concurrent::Callable< V > Class Template Reference	964
6.155.1 Detailed Description	964
6.155.2 Constructor & Destructor Documentation	964
6.155.2.1 ~Callable	964
6.155.3 Member Function Documentation	964
6.155.3.1 call	964
6.156 decaf::util::concurrent::CancellationException Class Reference	965
6.156.1 Constructor & Destructor Documentation	965
6.156.1.1 CancellationException	965
6.156.1.2 CancellationException	965
6.156.1.3 CancellationException	966
6.156.1.4 CancellationException	966
6.156.1.5 CancellationException	966
6.156.1.6 CancellationException	966
6.156.1.7 ~CancellationException	967
6.156.2 Member Function Documentation	967
6.156.2.1 clone	967
6.157 decaf::security::cert::Certificate Class Reference	968
6.157.1 Detailed Description	968
6.157.2 Constructor & Destructor Documentation	969
6.157.2.1 ~Certificate	969
6.157.3 Member Function Documentation	969
6.157.3.1 equals	969
6.157.3.2 getEncoded	969
6.157.3.3 getPublicKey	969
6.157.3.4 getPublicKey	969
6.157.3.5 getType	970

6.157.3.6 toString	970
6.157.3.7 verify	970
6.157.3.8 verify	971
6.158decaf::security::cert::CertificateEncodingException Class Reference	972
6.158.1 Constructor & Destructor Documentation	972
6.158.1.1 CertificateEncodingException	972
6.158.1.2 CertificateEncodingException	972
6.158.1.3 CertificateEncodingException	972
6.158.1.4 CertificateEncodingException	973
6.158.1.5 ~CertificateEncodingException	973
6.158.2 Member Function Documentation	973
6.158.2.1 clone	973
6.159decaf::security::cert::CertificateException Class Reference	974
6.159.1 Constructor & Destructor Documentation	974
6.159.1.1 CertificateException	974
6.159.1.2 CertificateException	974
6.159.1.3 CertificateException	974
6.159.1.4 CertificateException	975
6.159.1.5 ~CertificateException	975
6.159.2 Member Function Documentation	975
6.159.2.1 clone	975
6.160decaf::security::cert::CertificateExpiredException Class Reference	976
6.160.1 Constructor & Destructor Documentation	976
6.160.1.1 CertificateExpiredException	976
6.160.1.2 CertificateExpiredException	976
6.160.1.3 CertificateExpiredException	976
6.160.1.4 CertificateExpiredException	977
6.160.1.5 ~CertificateExpiredException	977
6.160.2 Member Function Documentation	977
6.160.2.1 clone	977
6.161decaf::security::cert::CertificateNotYetValidException Class Reference	978
6.161.1 Constructor & Destructor Documentation	978
6.161.1.1 CertificateNotYetValidException	978
6.161.1.2 CertificateNotYetValidException	978
6.161.1.3 CertificateNotYetValidException	978
6.161.1.4 CertificateNotYetValidException	979

6.161.1.5 ~CertificateNotYetValidException	979
6.161.2 Member Function Documentation	979
6.161.2.1 clone	979
6.162decaf::security::cert::CertificateParsingException Class Reference	980
6.162.1 Constructor & Destructor Documentation	980
6.162.1.1 CertificateParsingException	980
6.162.1.2 CertificateParsingException	980
6.162.1.3 CertificateParsingException	980
6.162.1.4 CertificateParsingException	981
6.162.1.5 ~CertificateParsingException	981
6.162.2 Member Function Documentation	981
6.162.2.1 clone	981
6.163decaf::lang::Character Class Reference	982
6.163.1 Constructor & Destructor Documentation	984
6.163.1.1 Character	984
6.163.2 Member Function Documentation	984
6.163.2.1 byteValue	984
6.163.2.2 compareTo	984
6.163.2.3 compareTo	984
6.163.2.4 digit	985
6.163.2.5 doubleValue	985
6.163.2.6 equals	985
6.163.2.7 equals	985
6.163.2.8 float Value	985
6.163.2.9 int Value	986
6.163.2.10sDigit	986
6.163.2.11sISOControl	986
6.163.2.12sLetter	986
6.163.2.13sLetterOrDigit	986
6.163.2.14sLowerCase	986
6.163.2.15sUpperCase	986
6.163.2.16sWhitespace	987
6.163.2.17ongValue	987
6.163.2.18operator<	987
6.163.2.19operator<	987
6.163.2.20operator==	987

6.163.2.21	operator==	988
6.163.2.22	shortValue	988
6.163.2.23	toString	988
6.163.2.24	valueOf	988
6.163.3	Field Documentation	989
6.163.3.1	MAX_RADIX	989
6.163.3.2	MAX_VALUE	989
6.163.3.3	MIN_RADIX	989
6.163.3.4	MIN_VALUE	989
6.163.3.5	SIZE	989
6.164	decaf::internal::nio::CharArrayBuffer Class Reference	990
6.164.1	Constructor & Destructor Documentation	991
6.164.1.1	CharArrayBuffer	991
6.164.1.2	CharArrayBuffer	992
6.164.1.3	CharArrayBuffer	992
6.164.1.4	CharArrayBuffer	992
6.164.1.5	~CharArrayBuffer	993
6.164.2	Member Function Documentation	993
6.164.2.1	array	993
6.164.2.2	arrayOffset	993
6.164.2.3	asReadOnlyBuffer	993
6.164.2.4	compact	994
6.164.2.5	duplicate	994
6.164.2.6	get	994
6.164.2.7	get	995
6.164.2.8	hasArray	995
6.164.2.9	isReadOnly	995
6.164.2.10	put	995
6.164.2.11	put	996
6.164.2.12	setReadOnly	996
6.164.2.13	slice	996
6.164.2.14	subSequence	997
6.164.3	Field Documentation	997
6.164.3.1	_array	997
6.164.3.2	offset	997
6.164.3.3	readOnly	997

6.165decaf::nio::CharBuffer Class Reference	998
6.165.1 Detailed Description	1000
6.165.2 Constructor & Destructor Documentation	1001
6.165.2.1 CharBuffer	1001
6.165.2.2 ~CharBuffer	1001
6.165.3 Member Function Documentation	1001
6.165.3.1 allocate	1001
6.165.3.2 append	1001
6.165.3.3 append	1002
6.165.3.4 append	1002
6.165.3.5 array	1003
6.165.3.6 arrayOffset	1003
6.165.3.7 asReadOnlyBuffer	1003
6.165.3.8 charAt	1004
6.165.3.9 compact	1004
6.165.3.10compareTo	1005
6.165.3.11duplicate	1005
6.165.3.12equals	1005
6.165.3.13get	1005
6.165.3.14get	1006
6.165.3.15get	1006
6.165.3.16get	1006
6.165.3.17hasArray	1007
6.165.3.18length	1007
6.165.3.19operator<	1007
6.165.3.20operator==	1007
6.165.3.21put	1008
6.165.3.22put	1008
6.165.3.23put	1009
6.165.3.24put	1009
6.165.3.25put	1009
6.165.3.26put	1010
6.165.3.27put	1010
6.165.3.28read	1011
6.165.3.29slice	1011
6.165.3.30subSequence	1012

6.165.3.31toString	1012
6.165.3.32wrap	1012
6.165.3.33wrap	1013
6.166decaf::lang::CharSequence Class Reference	1014
6.166.1 Detailed Description	1014
6.166.2 Constructor & Destructor Documentation	1014
6.166.2.1 ~CharSequence	1014
6.166.3 Member Function Documentation	1014
6.166.3.1 charAt	1014
6.166.3.2 length	1015
6.166.3.3 subSequence	1015
6.166.3.4 toString	1015
6.167decaf::lang::exceptions::ClassCastException Class Reference	1016
6.167.1 Constructor & Destructor Documentation	1016
6.167.1.1 ClassCastException	1016
6.167.1.2 ClassCastException	1016
6.167.1.3 ClassCastException	1017
6.167.1.4 ClassCastException	1017
6.167.1.5 ClassCastException	1017
6.167.1.6 ClassCastException	1017
6.167.1.7 ~ClassCastException	1018
6.167.2 Member Function Documentation	1018
6.167.2.1 clone	1018
6.168decaf::io::Closeable Class Reference	1019
6.168.1 Detailed Description	1019
6.168.2 Constructor & Destructor Documentation	1019
6.168.2.1 ~Closeable	1019
6.168.3 Member Function Documentation	1019
6.168.3.1 close	1019
6.169cms::Closeable Class Reference	1021
6.169.1 Detailed Description	1021
6.169.2 Constructor & Destructor Documentation	1021
6.169.2.1 ~Closeable	1021
6.169.3 Member Function Documentation	1021
6.169.3.1 close	1021
6.170activemq::transport::failover::CloseTransportsTask Class Reference	1022

6.170.1 Constructor & Destructor Documentation	1022
6.170.1.1 CloseTransportsTask	1022
6.170.1.2 ~CloseTransportsTask	1022
6.170.2 Member Function Documentation	1022
6.170.2.1 add	1022
6.170.2.2 isPending	1022
6.170.2.3 iterate	1023
6.171activemq::cmsutil::CmsAccessor Class Reference	1024
6.171.1 Detailed Description	1025
6.171.2 Constructor & Destructor Documentation	1025
6.171.2.1 CmsAccessor	1025
6.171.2.2 ~CmsAccessor	1025
6.171.3 Member Function Documentation	1025
6.171.3.1 checkConnectionFactory	1025
6.171.3.2 createConnection	1025
6.171.3.3 createSession	1026
6.171.3.4 destroy	1026
6.171.3.5 getConnectionFactory	1026
6.171.3.6 getConnectionFactory	1026
6.171.3.7 getResourceLifecycleManager	1026
6.171.3.8 getResourceLifecycleManager	1026
6.171.3.9 getSessionAcknowledgeMode	1026
6.171.3.10init	1027
6.171.3.11setConnectionFactory	1027
6.171.3.12setSessionAcknowledgeMode	1027
6.172activemq::cmsutil::CmsDestinationAccessor Class Reference	1028
6.172.1 Detailed Description	1028
6.172.2 Constructor & Destructor Documentation	1029
6.172.2.1 CmsDestinationAccessor	1029
6.172.2.2 ~CmsDestinationAccessor	1029
6.172.3 Member Function Documentation	1029
6.172.3.1 checkDestinationResolver	1029
6.172.3.2 destroy	1029
6.172.3.3 getDestinationResolver	1029
6.172.3.4 getDestinationResolver	1029
6.172.3.5 init	1029

6.172.3.6 isPubSubDomain	1030
6.172.3.7 resolveDestinationName	1030
6.172.3.8 setDestinationResolver	1030
6.172.3.9 setPubSubDomain	1030
6.173cms::CMSException Class Reference	1031
6.173.1 Detailed Description	1031
6.173.2 Constructor & Destructor Documentation	1032
6.173.2.1 CMSException	1032
6.173.2.2 CMSException	1032
6.173.2.3 CMSException	1032
6.173.2.4 CMSException	1032
6.173.2.5 ~CMSException	1032
6.173.3 Member Function Documentation	1032
6.173.3.1 getCause	1032
6.173.3.2 getMessage	1032
6.173.3.3 getStackTrace	1032
6.173.3.4 getStackTraceString	1033
6.173.3.5 printStackTrace	1033
6.173.3.6 printStackTrace	1033
6.173.3.7 setMark	1033
6.173.3.8 what	1033
6.174activemq::util::CMSExceptionSupport Class Reference	1034
6.174.1 Constructor & Destructor Documentation	1034
6.174.1.1 ~CMSExceptionSupport	1034
6.174.2 Member Function Documentation	1034
6.174.2.1 create	1034
6.174.2.2 create	1034
6.174.2.3 createMessageEOFException	1034
6.174.2.4 createMessageFormatException	1034
6.175cms::CMSProperties Class Reference	1036
6.175.1 Detailed Description	1037
6.175.2 Constructor & Destructor Documentation	1037
6.175.2.1 ~CMSProperties	1037
6.175.3 Member Function Documentation	1037
6.175.3.1 clear	1037
6.175.3.2 clone	1037

6.175.3.3 copy	1037
6.175.3.4 getProperty	1037
6.175.3.5 getProperty	1038
6.175.3.6 hasProperty	1038
6.175.3.7 isEmpty	1038
6.175.3.8 remove	1038
6.175.3.9 setProperty	1039
6.175.3.10oArray	1039
6.175.3.11toString	1039
6.176cms::CMSSecurityException Class Reference	1040
6.176.1 Detailed Description	1040
6.176.2 Constructor & Destructor Documentation	1040
6.176.2.1 CMSSecurityException	1040
6.176.2.2 CMSSecurityException	1040
6.176.2.3 CMSSecurityException	1040
6.176.2.4 CMSSecurityException	1040
6.176.2.5 ~CMSSecurityException	1040
6.177activemq::cmsutil::CmsTemplate Class Reference	1041
6.177.1 Detailed Description	1044
6.177.2 Constructor & Destructor Documentation	1044
6.177.2.1 CmsTemplate	1044
6.177.2.2 CmsTemplate	1044
6.177.2.3 ~CmsTemplate	1044
6.177.3 Member Function Documentation	1044
6.177.3.1 destroy	1044
6.177.3.2 execute	1045
6.177.3.3 execute	1045
6.177.3.4 execute	1045
6.177.3.5 execute	1045
6.177.3.6 getDefaultDestination	1046
6.177.3.7 getDefaultDestination	1046
6.177.3.8 getDefaultDestinationName	1046
6.177.3.9 getDeliveryMode	1046
6.177.3.10getPriority	1046
6.177.3.11getReceiveTimeout	1046
6.177.3.12getTimeToLive	1046

6.177.3.13	nit	1047
6.177.3.14	sExplicitQosEnabled	1047
6.177.3.15	sMessageIdEnabled	1047
6.177.3.16	sMessageTimestampEnabled	1047
6.177.3.17	sNoLocal	1047
6.177.3.18	receive	1047
6.177.3.19	receive	1048
6.177.3.20	receive	1048
6.177.3.21	receiveSelected	1048
6.177.3.22	receiveSelected	1049
6.177.3.23	receiveSelected	1049
6.177.3.24	end	1049
6.177.3.25	end	1050
6.177.3.26	end	1050
6.177.3.27	setDefaultDestination	1050
6.177.3.28	setDefaultDestinationName	1050
6.177.3.29	setDeliveryMode	1051
6.177.3.30	setDeliveryPersistent	1051
6.177.3.31	setExplicitQosEnabled	1051
6.177.3.32	setMessageIdEnabled	1052
6.177.3.33	setMessageTimestampEnabled	1052
6.177.3.34	setNoLocal	1052
6.177.3.35	setPriority	1052
6.177.3.36	setPubSubDomain	1052
6.177.3.37	setReceiveTimeout	1052
6.177.3.38	setTimeToLive	1052
6.177.4	Friends And Related Function Documentation	1053
6.177.4.1	ProducerExecutor	1053
6.177.4.2	ReceiveExecutor	1053
6.177.4.3	ResolveProducerExecutor	1053
6.177.4.4	ResolveReceiveExecutor	1053
6.177.4.5	SendExecutor	1053
6.177.5	Field Documentation	1053
6.177.5.1	DEFAULT_PRIORITY	1053
6.177.5.2	DEFAULT_TIME_TO_LIVE	1053
6.177.5.3	RECEIVE_TIMEOUT_INDEFINITE_WAIT	1053

6.177.5.4 RECEIVE_TIMEOUT_NO_WAIT	1053
6.178decaf::util::Collection< E > Class Template Reference	1054
6.178.1 Detailed Description	1055
6.178.2 Constructor & Destructor Documentation	1056
6.178.2.1 ~Collection	1056
6.178.3 Member Function Documentation	1056
6.178.3.1 add	1056
6.178.3.2 addAll	1057
6.178.3.3 clear	1057
6.178.3.4 contains	1058
6.178.3.5 containsAll	1058
6.178.3.6 equals	1059
6.178.3.7 isEmpty	1059
6.178.3.8 remove	1060
6.178.3.9 removeAll	1060
6.178.3.10 retainAll	1061
6.178.3.11 size	1062
6.178.3.12 toArray	1062
6.179activemq::commands::Command Class Reference	1063
6.179.1 Constructor & Destructor Documentation	1064
6.179.1.1 ~Command	1064
6.179.2 Member Function Documentation	1064
6.179.2.1 getCommandId	1064
6.179.2.2 isBrokerInfo	1064
6.179.2.3 isConnectionInfo	1064
6.179.2.4 isConsumerInfo	1064
6.179.2.5 isKeepAliveInfo	1064
6.179.2.6 isMessage	1064
6.179.2.7 isMessageAck	1065
6.179.2.8 isMessageDispatch	1065
6.179.2.9 isMessageDispatchNotification	1065
6.179.2.10 isProducerAck	1065
6.179.2.11 isProducerInfo	1065
6.179.2.12 isRemoveInfo	1065
6.179.2.13 isRemoveSubscriptionInfo	1065
6.179.2.14 isResponse	1065

6.179.2.15sResponseRequired	1066
6.179.2.16sShutdownInfo	1066
6.179.2.17sTransactionInfo	1066
6.179.2.18sWireFormatInfo	1066
6.179.2.19set CommandId	1066
6.179.2.20set ResponseRequired	1066
6.179.2.21toString	1067
6.179.2.22visit	1067
6.180activemq::state::CommandVisitor Class Reference	1069
6.180.1 Detailed Description	1070
6.180.2 Constructor & Destructor Documentation	1071
6.180.2.1 ~CommandVisitor	1071
6.180.3 Member Function Documentation	1071
6.180.3.1 processBeginTransaction	1071
6.180.3.2 processBrokerError	1071
6.180.3.3 processBrokerInfo	1071
6.180.3.4 processCommitTransactionOnePhase	1071
6.180.3.5 processCommitTransactionTwoPhase	1071
6.180.3.6 processConnectionControl	1072
6.180.3.7 processConnectionError	1072
6.180.3.8 processConnectionInfo	1072
6.180.3.9 processConsumerControl	1072
6.180.3.10processConsumerInfo	1072
6.180.3.11processControlCommand	1072
6.180.3.12processDestinationInfo	1072
6.180.3.13processEndTransaction	1072
6.180.3.14processFlushCommand	1073
6.180.3.15processForgetTransaction	1073
6.180.3.16processKeepAliveInfo	1073
6.180.3.17processMessage	1073
6.180.3.18processMessageAck	1073
6.180.3.19processMessageDispatch	1073
6.180.3.20processMessageDispatchNotification	1073
6.180.3.21processMessagePull	1073
6.180.3.22processPrepareTransaction	1073
6.180.3.23processProducerAck	1074

6.180.3.24	processProducerInfo	1074
6.180.3.25	processRecoverTransactions	1074
6.180.3.26	processRemoveConnection	1074
6.180.3.27	processRemoveConsumer	1074
6.180.3.28	processRemoveDestination	1074
6.180.3.29	processRemoveInfo	1074
6.180.3.30	processRemoveProducer	1074
6.180.3.31	processRemoveSession	1075
6.180.3.32	processRemoveSubscriptionInfo	1075
6.180.3.33	processReplayCommand	1075
6.180.3.34	processResponse	1075
6.180.3.35	processRollbackTransaction	1075
6.180.3.36	processSessionInfo	1075
6.180.3.37	processShutdownInfo	1075
6.180.3.38	processTransactionInfo	1075
6.180.3.39	processWireFormat	1076
6.181	activemq::state::CommandVisitorAdapter Class Reference	1077
6.181.1	Detailed Description	1079
6.181.2	Constructor & Destructor Documentation	1080
6.181.2.1	~CommandVisitorAdapter	1080
6.181.3	Member Function Documentation	1080
6.181.3.1	processBeginTransaction	1080
6.181.3.2	processBrokerError	1080
6.181.3.3	processBrokerInfo	1080
6.181.3.4	processCommitTransactionOnePhase	1080
6.181.3.5	processCommitTransactionTwoPhase	1080
6.181.3.6	processConnectionControl	1080
6.181.3.7	processConnectionError	1080
6.181.3.8	processConnectionInfo	1080
6.181.3.9	processConsumerControl	1080
6.181.3.10	processConsumerInfo	1080
6.181.3.11	processControlCommand	1080
6.181.3.12	processDestinationInfo	1080
6.181.3.13	processEndTransaction	1080
6.181.3.14	processFlushCommand	1080
6.181.3.15	processForgetTransaction	1080

6.181.3.16	processKeepAliveInfo	1080
6.181.3.17	processMessage	1080
6.181.3.18	processMessageAck	1080
6.181.3.19	processMessageDispatch	1080
6.181.3.20	processMessageDispatchNotification	1080
6.181.3.21	processMessagePull	1080
6.181.3.22	processPrepareTransaction	1080
6.181.3.23	processProducerAck	1080
6.181.3.24	processProducerInfo	1080
6.181.3.25	processRecoverTransactions	1080
6.181.3.26	processRemoveConnection	1080
6.181.3.27	processRemoveConsumer	1080
6.181.3.28	processRemoveDestination	1080
6.181.3.29	processRemoveInfo	1080
6.181.3.30	processRemoveProducer	1081
6.181.3.31	processRemoveSession	1081
6.181.3.32	processRemoveSubscriptionInfo	1081
6.181.3.33	processReplayCommand	1081
6.181.3.34	processResponse	1081
6.181.3.35	processRollbackTransaction	1081
6.181.3.36	processSessionInfo	1081
6.181.3.37	processShutdownInfo	1081
6.181.3.38	processTransactionInfo	1081
6.181.3.39	processWireFormat	1082
6.182	decaf::lang::Comparable< T > Class Template Reference	1083
6.182.1	Detailed Description	1083
6.182.2	Constructor & Destructor Documentation	1083
6.182.2.1	~Comparable	1083
6.182.3	Member Function Documentation	1083
6.182.3.1	compareTo	1083
6.182.3.2	equals	1084
6.182.3.3	operator<	1084
6.182.3.4	operator==	1085
6.183	decaf::util::Comparator< T > Class Template Reference	1086
6.183.1	Detailed Description	1086
6.183.2	Constructor & Destructor Documentation	1086

6.183.2.1 ~Comparator	1086
6.183.3 Member Function Documentation	1086
6.183.3.1 compare	1086
6.183.3.2 operator()	1087
6.184activemq::util::CompositeData Class Reference	1088
6.184.1 Detailed Description	1088
6.184.2 Constructor & Destructor Documentation	1089
6.184.2.1 CompositeData	1089
6.184.2.2 ~CompositeData	1089
6.184.3 Member Function Documentation	1089
6.184.3.1 getComponents	1089
6.184.3.2 getComponents	1089
6.184.3.3 getFragment	1089
6.184.3.4 getHost	1089
6.184.3.5 getParameters	1089
6.184.3.6 getPath	1089
6.184.3.7 getScheme	1089
6.184.3.8 setComponents	1089
6.184.3.9 setFragment	1089
6.184.3.10setHost	1089
6.184.3.11setParameters	1089
6.184.3.12setPath	1089
6.184.3.13setScheme	1089
6.184.3.14oURI	1089
6.185activemq::threads::CompositeTask Class Reference	1090
6.185.1 Detailed Description	1090
6.185.2 Constructor & Destructor Documentation	1090
6.185.2.1 ~CompositeTask	1090
6.185.3 Member Function Documentation	1090
6.185.3.1 isPending	1090
6.186activemq::threads::CompositeTaskRunner Class Reference	1092
6.186.1 Detailed Description	1092
6.186.2 Constructor & Destructor Documentation	1093
6.186.2.1 CompositeTaskRunner	1093
6.186.2.2 ~CompositeTaskRunner	1093
6.186.3 Member Function Documentation	1093

6.186.3.1	addTask	1093
6.186.3.2	iterate	1093
6.186.3.3	removeTask	1093
6.186.3.4	run	1093
6.186.3.5	shutdown	1093
6.186.3.6	shutdown	1094
6.186.3.7	wakeup	1094
6.187	activemq::transport::CompositeTransport Class Reference	1095
6.187.1	Detailed Description	1095
6.187.2	Constructor & Destructor Documentation	1095
6.187.2.1	~CompositeTransport	1095
6.187.3	Member Function Documentation	1095
6.187.3.1	addURI	1095
6.187.3.2	removeURI	1096
6.188	decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference	1097
6.188.1	Detailed Description	1097
6.188.2	Constructor & Destructor Documentation	1098
6.188.2.1	~ConcurrentMap	1098
6.188.3	Member Function Documentation	1098
6.188.3.1	putIfAbsent	1098
6.188.3.2	remove	1099
6.188.3.3	replace	1099
6.188.3.4	replace	1100
6.189	decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference	1102
6.189.1	Detailed Description	1105
6.189.2	Constructor & Destructor Documentation	1106
6.189.2.1	ConcurrentStlMap	1106
6.189.2.2	ConcurrentStlMap	1106
6.189.2.3	ConcurrentStlMap	1106
6.189.2.4	~ConcurrentStlMap	1106
6.189.3	Member Function Documentation	1106
6.189.3.1	clear	1106
6.189.3.2	containsKey	1107
6.189.3.3	containsValue	1107
6.189.3.4	copy	1107

6.189.3.5 copy	1108
6.189.3.6 equals	1108
6.189.3.7 equals	1108
6.189.3.8 get	1108
6.189.3.9 get	1109
6.189.3.10 isEmpty	1109
6.189.3.11 keySet	1109
6.189.3.12 lock	1110
6.189.3.13 notify	1110
6.189.3.14 notifyAll	1110
6.189.3.15 put	1111
6.189.3.16 putAll	1111
6.189.3.17 putAll	1111
6.189.3.18 putIfAbsent	1112
6.189.3.19 remove	1112
6.189.3.20 remove	1113
6.189.3.21 replace	1113
6.189.3.22 replace	1114
6.189.3.23 size	1114
6.189.3.24 tryLock	1115
6.189.3.25 unlock	1115
6.189.3.26 values	1115
6.189.3.27 wait	1115
6.189.3.28 wait	1116
6.189.3.29 wait	1116
6.190 decaf::util::concurrent::locks::Condition Class Reference	1118
6.190.1 Detailed Description	1118
6.190.2 Constructor & Destructor Documentation	1120
6.190.2.1 ~Condition	1120
6.190.3 Member Function Documentation	1120
6.190.3.1 await	1120
6.190.3.2 await	1120
6.190.3.3 awaitNanos	1121
6.190.3.4 awaitUninterruptibly	1122
6.190.3.5 awaitUntil	1123
6.190.3.6 signal	1123

6.190.3.7 signalAll	1123
6.191decaf::util::concurrent::ConditionHandle Class Reference	1124
6.191.1 Constructor & Destructor Documentation	1124
6.191.1.1 ConditionHandle	1124
6.191.1.2 ~ConditionHandle	1124
6.191.1.3 ConditionHandle	1124
6.191.1.4 ~ConditionHandle	1124
6.191.2 Field Documentation	1124
6.191.2.1 condition	1124
6.191.2.2 criticalSection	1124
6.191.2.3 generation	1124
6.191.2.4 mutex	1124
6.191.2.5 numWaiting	1124
6.191.2.6 numWake	1124
6.191.2.7 semaphore	1124
6.192decaf::internal::util::concurrent::ConditionImpl Class Reference	1126
6.192.1 Member Function Documentation	1126
6.192.1.1 create	1126
6.192.1.2 destroy	1126
6.192.1.3 notify	1127
6.192.1.4 notifyAll	1127
6.192.1.5 wait	1127
6.192.1.6 wait	1127
6.193decaf::net::ConnectException Class Reference	1128
6.193.1 Constructor & Destructor Documentation	1128
6.193.1.1 ConnectException	1128
6.193.1.2 ConnectException	1128
6.193.1.3 ConnectException	1129
6.193.1.4 ConnectException	1129
6.193.1.5 ConnectException	1129
6.193.1.6 ConnectException	1129
6.193.1.7 ~ConnectException	1130
6.193.2 Member Function Documentation	1130
6.193.2.1 clone	1130
6.194cms::Connection Class Reference	1131
6.194.1 Detailed Description	1131

6.194.2 Constructor & Destructor Documentation	1132
6.194.2.1 ~Connection	1132
6.194.3 Member Function Documentation	1132
6.194.3.1 close	1132
6.194.3.2 createSession	1132
6.194.3.3 createSession	1133
6.194.3.4 getClientID	1133
6.194.3.5 getExceptionListener	1133
6.194.3.6 getMetaData	1133
6.194.3.7 setExceptionListener	1134
6.195activemq::commands::ConnectionControl Class Reference	1135
6.195.1 Constructor & Destructor Documentation	1136
6.195.1.1 ConnectionControl	1136
6.195.1.2 ConnectionControl	1136
6.195.1.3 ~ConnectionControl	1136
6.195.2 Member Function Documentation	1136
6.195.2.1 cloneDataStructure	1136
6.195.2.2 copyDataStructure	1136
6.195.2.3 equals	1137
6.195.2.4 getDataStructureType	1137
6.195.2.5 isClose	1138
6.195.2.6 isExit	1138
6.195.2.7 isFaultTolerant	1138
6.195.2.8 isResume	1138
6.195.2.9 isSuspend	1138
6.195.2.10operator=	1138
6.195.2.11setClose	1138
6.195.2.12setExit	1138
6.195.2.13setFaultTolerant	1138
6.195.2.14setResume	1138
6.195.2.15setSuspend	1138
6.195.2.16toString	1138
6.195.2.17visit	1138
6.195.3 Field Documentation	1139
6.195.3.1 close	1139
6.195.3.2 exit	1139

6.195.3.3	faultTolerant	1139
6.195.3.4	ID_CONNECTIONCONTROL	1139
6.195.3.5	resume	1139
6.195.3.6	suspend	1139
6.196	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class	
	Reference	1140
6.196.1	Detailed Description	1140
6.196.2	Constructor & Destructor Documentation	1141
6.196.2.1	ConnectionControlMarshaller	1141
6.196.2.2	~ConnectionControlMarshaller	1141
6.196.3	Member Function Documentation	1141
6.196.3.1	createObject	1141
6.196.3.2	getDataStructureType	1141
6.196.3.3	looseMarshal	1141
6.196.3.4	looseUnmarshal	1142
6.196.3.5	tightMarshal1	1142
6.196.3.6	tightMarshal2	1142
6.196.3.7	tightUnmarshal	1143
6.197	activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class	
	Reference	1144
6.197.1	Detailed Description	1144
6.197.2	Constructor & Destructor Documentation	1145
6.197.2.1	ConnectionControlMarshaller	1145
6.197.2.2	~ConnectionControlMarshaller	1145
6.197.3	Member Function Documentation	1145
6.197.3.1	createObject	1145
6.197.3.2	getDataStructureType	1145
6.197.3.3	looseMarshal	1145
6.197.3.4	looseUnmarshal	1146
6.197.3.5	tightMarshal1	1146
6.197.3.6	tightMarshal2	1146
6.197.3.7	tightUnmarshal	1147
6.198	activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class	
	Reference	1148
6.198.1	Detailed Description	1148
6.198.2	Constructor & Destructor Documentation	1149
6.198.2.1	ConnectionControlMarshaller	1149

6.198.2.2 ~ConnectionControlMarshaller	1149
6.198.3 Member Function Documentation	1149
6.198.3.1 createObject	1149
6.198.3.2 getDataStructureType	1149
6.198.3.3 looseMarshal	1149
6.198.3.4 looseUnmarshal	1150
6.198.3.5 tightMarshal1	1150
6.198.3.6 tightMarshal2	1150
6.198.3.7 tightUnmarshal	1151
6.199activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class	
Reference	1152
6.199.1 Detailed Description	1152
6.199.2 Constructor & Destructor Documentation	1153
6.199.2.1 ConnectionControlMarshaller	1153
6.199.2.2 ~ConnectionControlMarshaller	1153
6.199.3 Member Function Documentation	1153
6.199.3.1 createObject	1153
6.199.3.2 getDataStructureType	1153
6.199.3.3 looseMarshal	1153
6.199.3.4 looseUnmarshal	1154
6.199.3.5 tightMarshal1	1154
6.199.3.6 tightMarshal2	1154
6.199.3.7 tightUnmarshal	1155
6.200activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class	
Reference	1156
6.200.1 Detailed Description	1156
6.200.2 Constructor & Destructor Documentation	1157
6.200.2.1 ConnectionControlMarshaller	1157
6.200.2.2 ~ConnectionControlMarshaller	1157
6.200.3 Member Function Documentation	1157
6.200.3.1 createObject	1157
6.200.3.2 getDataStructureType	1157
6.200.3.3 looseMarshal	1157
6.200.3.4 looseUnmarshal	1158
6.200.3.5 tightMarshal1	1158
6.200.3.6 tightMarshal2	1158
6.200.3.7 tightUnmarshal	1159

6.201activemq::commands::ConnectionError Class Reference	1160
6.201.1 Constructor & Destructor Documentation	1161
6.201.1.1 ConnectionError	1161
6.201.1.2 ConnectionError	1161
6.201.1.3 ~ConnectionError	1161
6.201.2 Member Function Documentation	1161
6.201.2.1 cloneDataStructure	1161
6.201.2.2 copyDataStructure	1161
6.201.2.3 equals	1161
6.201.2.4 getConnectionId	1162
6.201.2.5 getConnectionId	1162
6.201.2.6 getDataStructureType	1162
6.201.2.7 getException	1162
6.201.2.8 getException	1162
6.201.2.9 operator=	1162
6.201.2.10 setConnectionId	1162
6.201.2.11 setException	1162
6.201.2.12 toString	1162
6.201.2.13 visit	1163
6.201.3 Field Documentation	1163
6.201.3.1 connectionId	1163
6.201.3.2 exception	1163
6.201.3.3 ID_CONNECTIONERROR	1163
6.202activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference	1164
6.202.1 Detailed Description	1164
6.202.2 Constructor & Destructor Documentation	1165
6.202.2.1 ConnectionErrorMarshaller	1165
6.202.2.2 ~ConnectionErrorMarshaller	1165
6.202.3 Member Function Documentation	1165
6.202.3.1 createObject	1165
6.202.3.2 getDataStructureType	1165
6.202.3.3 looseMarshal	1165
6.202.3.4 looseUnmarshal	1166
6.202.3.5 tightMarshal1	1166
6.202.3.6 tightMarshal2	1166
6.202.3.7 tightUnmarshal	1167

6.203	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	Class	
	Reference		1168
6.203.1	Detailed Description		1168
6.203.2	Constructor & Destructor Documentation		1169
6.203.2.1	ConnectionErrorMarshaller		1169
6.203.2.2	~ConnectionErrorMarshaller		1169
6.203.3	Member Function Documentation		1169
6.203.3.1	createObject		1169
6.203.3.2	getDataStructureType		1169
6.203.3.3	looseMarshal		1169
6.203.3.4	looseUnmarshal		1170
6.203.3.5	tightMarshal1		1170
6.203.3.6	tightMarshal2		1170
6.203.3.7	tightUnmarshal		1171
6.204	activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	Class	
	Reference		1172
6.204.1	Detailed Description		1172
6.204.2	Constructor & Destructor Documentation		1173
6.204.2.1	ConnectionErrorMarshaller		1173
6.204.2.2	~ConnectionErrorMarshaller		1173
6.204.3	Member Function Documentation		1173
6.204.3.1	createObject		1173
6.204.3.2	getDataStructureType		1173
6.204.3.3	looseMarshal		1173
6.204.3.4	looseUnmarshal		1174
6.204.3.5	tightMarshal1		1174
6.204.3.6	tightMarshal2		1174
6.204.3.7	tightUnmarshal		1175
6.205	activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	Class	
	Reference		1176
6.205.1	Detailed Description		1176
6.205.2	Constructor & Destructor Documentation		1177
6.205.2.1	ConnectionErrorMarshaller		1177
6.205.2.2	~ConnectionErrorMarshaller		1177
6.205.3	Member Function Documentation		1177
6.205.3.1	createObject		1177
6.205.3.2	getDataStructureType		1177

6.205.3.3	looseMarshal	1177
6.205.3.4	looseUnmarshal	1178
6.205.3.5	tightMarshal1	1178
6.205.3.6	tightMarshal2	1178
6.205.3.7	tightUnmarshal	1179
6.206	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference	1180
6.206.1	Detailed Description	1180
6.206.2	Constructor & Destructor Documentation	1181
6.206.2.1	ConnectionErrorMarshaller	1181
6.206.2.2	~ConnectionErrorMarshaller	1181
6.206.3	Member Function Documentation	1181
6.206.3.1	createObject	1181
6.206.3.2	getDataStructureType	1181
6.206.3.3	looseMarshal	1181
6.206.3.4	looseUnmarshal	1182
6.206.3.5	tightMarshal1	1182
6.206.3.6	tightMarshal2	1182
6.206.3.7	tightUnmarshal	1183
6.207	cms::ConnectionFactory Class Reference	1184
6.207.1	Detailed Description	1184
6.207.2	Constructor & Destructor Documentation	1185
6.207.2.1	~ConnectionFactory	1185
6.207.3	Member Function Documentation	1185
6.207.3.1	createCMSConnectionFactory	1185
6.207.3.2	createConnection	1185
6.207.3.3	createConnection	1186
6.207.3.4	createConnection	1186
6.208	activemq::commands::ConnectionId Class Reference	1187
6.208.1	Member Typedef Documentation	1188
6.208.1.1	COMPARATOR	1188
6.208.2	Constructor & Destructor Documentation	1188
6.208.2.1	ConnectionId	1188
6.208.2.2	ConnectionId	1188
6.208.2.3	ConnectionId	1188
6.208.2.4	ConnectionId	1188
6.208.2.5	ConnectionId	1188

6.208.2.6 ~ConnectionId	1188
6.208.3 Member Function Documentation	1188
6.208.3.1 cloneDataStructure	1188
6.208.3.2 compareTo	1188
6.208.3.3 copyDataStructure	1188
6.208.3.4 equals	1189
6.208.3.5 equals	1189
6.208.3.6 getDataStructureType	1189
6.208.3.7 getValue	1189
6.208.3.8 getValue	1189
6.208.3.9 operator<	1189
6.208.3.10 operator=	1189
6.208.3.11 operator==	1189
6.208.3.12 setValue	1189
6.208.3.13 toString	1189
6.208.4 Field Documentation	1190
6.208.4.1 ID_CONNECTIONID	1190
6.208.4.2 value	1190
6.209activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference	1191
6.209.1 Detailed Description	1191
6.209.2 Constructor & Destructor Documentation	1192
6.209.2.1 ConnectionIdMarshaller	1192
6.209.2.2 ~ConnectionIdMarshaller	1192
6.209.3 Member Function Documentation	1192
6.209.3.1 createObject	1192
6.209.3.2 getDataStructureType	1192
6.209.3.3 looseMarshal	1192
6.209.3.4 looseUnmarshal	1193
6.209.3.5 tightMarshal1	1193
6.209.3.6 tightMarshal2	1193
6.209.3.7 tightUnmarshal	1194
6.210activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference	1195
6.210.1 Detailed Description	1195
6.210.2 Constructor & Destructor Documentation	1196
6.210.2.1 ConnectionIdMarshaller	1196

6.210.2.2	~ConnectionIdMarshaller	1196
6.210.3	Member Function Documentation	1196
6.210.3.1	createObject	1196
6.210.3.2	getDataStructureType	1196
6.210.3.3	looseMarshal	1196
6.210.3.4	looseUnmarshal	1197
6.210.3.5	tightMarshal1	1197
6.210.3.6	tightMarshal2	1197
6.210.3.7	tightUnmarshal	1198
6.211	activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference	1199
6.211.1	Detailed Description	1199
6.211.2	Constructor & Destructor Documentation	1200
6.211.2.1	ConnectionIdMarshaller	1200
6.211.2.2	~ConnectionIdMarshaller	1200
6.211.3	Member Function Documentation	1200
6.211.3.1	createObject	1200
6.211.3.2	getDataStructureType	1200
6.211.3.3	looseMarshal	1200
6.211.3.4	looseUnmarshal	1201
6.211.3.5	tightMarshal1	1201
6.211.3.6	tightMarshal2	1201
6.211.3.7	tightUnmarshal	1202
6.212	activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference	1203
6.212.1	Detailed Description	1203
6.212.2	Constructor & Destructor Documentation	1204
6.212.2.1	ConnectionIdMarshaller	1204
6.212.2.2	~ConnectionIdMarshaller	1204
6.212.3	Member Function Documentation	1204
6.212.3.1	createObject	1204
6.212.3.2	getDataStructureType	1204
6.212.3.3	looseMarshal	1204
6.212.3.4	looseUnmarshal	1205
6.212.3.5	tightMarshal1	1205
6.212.3.6	tightMarshal2	1205
6.212.3.7	tightUnmarshal	1206

6.213activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Reference	1207
6.213.1 Detailed Description	1207
6.213.2 Constructor & Destructor Documentation	1208
6.213.2.1 ConnectionIdMarshaller	1208
6.213.2.2 ~ConnectionIdMarshaller	1208
6.213.3 Member Function Documentation	1208
6.213.3.1 createObject	1208
6.213.3.2 getDataStructureType	1208
6.213.3.3 looseMarshal	1208
6.213.3.4 looseUnmarshal	1209
6.213.3.5 tightMarshal1	1209
6.213.3.6 tightMarshal2	1209
6.213.3.7 tightUnmarshal	1210
6.214activemq::commands::ConnectionInfo Class Reference	1211
6.214.1 Constructor & Destructor Documentation	1212
6.214.1.1 ConnectionInfo	1212
6.214.1.2 ConnectionInfo	1212
6.214.1.3 ~ConnectionInfo	1212
6.214.2 Member Function Documentation	1212
6.214.2.1 cloneDataStructure	1212
6.214.2.2 copyDataStructure	1213
6.214.2.3 equals	1213
6.214.2.4 getBrokerPath	1213
6.214.2.5 getBrokerPath	1213
6.214.2.6 getClientId	1213
6.214.2.7 getClientId	1213
6.214.2.8 getConnectionId	1213
6.214.2.9 getConnectionId	1213
6.214.2.10getDataStructureType	1213
6.214.2.11getPassword	1214
6.214.2.12getPassword	1214
6.214.2.13getUserName	1214
6.214.2.14getUserName	1214
6.214.2.15sBrokerMasterConnector	1214
6.214.2.16sClientMaster	1214
6.214.2.17sConnectionInfo	1214

6.214.2.18	manageable	1215
6.214.2.19	operator=	1215
6.214.2.20	setBrokerMasterConnector	1215
6.214.2.21	setBrokerPath	1215
6.214.2.22	setClientId	1215
6.214.2.23	setClientMaster	1215
6.214.2.24	setConnectionId	1215
6.214.2.25	setManageable	1215
6.214.2.26	setPassword	1215
6.214.2.27	setUserName	1215
6.214.2.28	toString	1215
6.214.2.29	visit	1215
6.214.3	Field Documentation	1216
6.214.3.1	brokerMasterConnector	1216
6.214.3.2	brokerPath	1216
6.214.3.3	clientId	1216
6.214.3.4	clientMaster	1216
6.214.3.5	connectionId	1216
6.214.3.6	ID_CONNECTIONINFO	1216
6.214.3.7	manageable	1216
6.214.3.8	password	1216
6.214.3.9	userName	1216
6.215	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference	1217
6.215.1	Detailed Description	1217
6.215.2	Constructor & Destructor Documentation	1218
6.215.2.1	ConnectionInfoMarshaller	1218
6.215.2.2	~ConnectionInfoMarshaller	1218
6.215.3	Member Function Documentation	1218
6.215.3.1	createObject	1218
6.215.3.2	getDataStructureType	1218
6.215.3.3	looseMarshal	1218
6.215.3.4	looseUnmarshal	1219
6.215.3.5	tightMarshal1	1219
6.215.3.6	tightMarshal2	1219
6.215.3.7	tightUnmarshal	1220

6.216	activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference	1221
6.216.1	Detailed Description	1221
6.216.2	Constructor & Destructor Documentation	1222
6.216.2.1	ConnectionInfoMarshaller	1222
6.216.2.2	~ConnectionInfoMarshaller	1222
6.216.3	Member Function Documentation	1222
6.216.3.1	createObject	1222
6.216.3.2	getDataStructureType	1222
6.216.3.3	looseMarshal	1222
6.216.3.4	looseUnmarshal	1223
6.216.3.5	tightMarshal1	1223
6.216.3.6	tightMarshal2	1223
6.216.3.7	tightUnmarshal	1224
6.217	activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference	1225
6.217.1	Detailed Description	1225
6.217.2	Constructor & Destructor Documentation	1226
6.217.2.1	ConnectionInfoMarshaller	1226
6.217.2.2	~ConnectionInfoMarshaller	1226
6.217.3	Member Function Documentation	1226
6.217.3.1	createObject	1226
6.217.3.2	getDataStructureType	1226
6.217.3.3	looseMarshal	1226
6.217.3.4	looseUnmarshal	1227
6.217.3.5	tightMarshal1	1227
6.217.3.6	tightMarshal2	1227
6.217.3.7	tightUnmarshal	1228
6.218	activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference	1229
6.218.1	Detailed Description	1229
6.218.2	Constructor & Destructor Documentation	1230
6.218.2.1	ConnectionInfoMarshaller	1230
6.218.2.2	~ConnectionInfoMarshaller	1230
6.218.3	Member Function Documentation	1230
6.218.3.1	createObject	1230
6.218.3.2	getDataStructureType	1230

6.218.3.3 looseMarshal	1230
6.218.3.4 looseUnmarshal	1231
6.218.3.5 tightMarshal1	1231
6.218.3.6 tightMarshal2	1231
6.218.3.7 tightUnmarshal	1232
6.219activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference	1233
6.219.1 Detailed Description	1233
6.219.2 Constructor & Destructor Documentation	1234
6.219.2.1 ConnectionInfoMarshaller	1234
6.219.2.2 ~ConnectionInfoMarshaller	1234
6.219.3 Member Function Documentation	1234
6.219.3.1 createObject	1234
6.219.3.2 getDataStructureType	1234
6.219.3.3 looseMarshal	1234
6.219.3.4 looseUnmarshal	1235
6.219.3.5 tightMarshal1	1235
6.219.3.6 tightMarshal2	1235
6.219.3.7 tightUnmarshal	1236
6.220cms::ConnectionMetaData Class Reference	1237
6.220.1 Detailed Description	1237
6.220.2 Constructor & Destructor Documentation	1238
6.220.2.1 ~ConnectionMetaData	1238
6.220.3 Member Function Documentation	1238
6.220.3.1 getCMSMajorVersion	1238
6.220.3.2 getCMSMinorVersion	1238
6.220.3.3 getCMSProviderName	1238
6.220.3.4 getCMSVersion	1239
6.220.3.5 getCMSXPropertyNames	1239
6.220.3.6 getProviderMajorVersion	1239
6.220.3.7 getProviderMinorVersion	1239
6.220.3.8 getProviderVersion	1240
6.221activemq::state::ConnectionState Class Reference	1241
6.221.1 Constructor & Destructor Documentation	1242
6.221.1.1 ConnectionState	1242
6.221.1.2 ~ConnectionState	1242
6.221.2 Member Function Documentation	1242

6.221.2.1 addSession	1242
6.221.2.2 addTempDestination	1242
6.221.2.3 addTransactionState	1242
6.221.2.4 checkShutdown	1242
6.221.2.5 getInfo	1242
6.221.2.6 getSessionState	1242
6.221.2.7 getSessionStates	1242
6.221.2.8 getTempDesinations	1242
6.221.2.9 getTransactionState	1242
6.221.2.10getTransactionStates	1242
6.221.2.11removeSession	1242
6.221.2.12removeTempDestination	1242
6.221.2.13removeTransactionState	1243
6.221.2.14reset	1243
6.221.2.15shutdown	1243
6.221.2.16oString	1243
6.222activemq::state::ConnectionStateTracker Class Reference	1244
6.222.1 Constructor & Destructor Documentation	1246
6.222.1.1 ConnectionStateTracker	1246
6.222.1.2 ~ConnectionStateTracker	1246
6.222.2 Member Function Documentation	1246
6.222.2.1 getMaxCacheSize	1246
6.222.2.2 isRestoreConsumers	1246
6.222.2.3 isRestoreProducers	1246
6.222.2.4 isRestoreSessions	1246
6.222.2.5 isRestoreTransaction	1246
6.222.2.6 isTrackMessages	1246
6.222.2.7 isTrackTransactions	1246
6.222.2.8 processBeginTransaction	1246
6.222.2.9 processCommitTransactionOnePhase	1246
6.222.2.10processCommitTransactionTwoPhase	1246
6.222.2.11processConnectionInfo	1247
6.222.2.12processConsumerInfo	1247
6.222.2.13processDestinationInfo	1247
6.222.2.14processEndTransaction	1247
6.222.2.15processMessage	1247

6.222.2.16	processMessageAck	1247
6.222.2.17	processPrepareTransaction	1247
6.222.2.18	processProducerInfo	1248
6.222.2.19	processRemoveConnection	1248
6.222.2.20	processRemoveConsumer	1248
6.222.2.21	processRemoveDestination	1248
6.222.2.22	processRemoveProducer	1248
6.222.2.23	processRemoveSession	1248
6.222.2.24	processRollbackTransaction	1248
6.222.2.25	processSessionInfo	1249
6.222.2.26	restore	1249
6.222.2.27	setMaxCacheSize	1249
6.222.2.28	setRestoreConsumers	1249
6.222.2.29	setRestoreProducers	1249
6.222.2.30	setRestoreSessions	1249
6.222.2.31	setRestoreTransaction	1249
6.222.2.32	setTrackMessages	1249
6.222.2.33	setTrackTransactions	1249
6.222.2.34	track	1249
6.222.2.35	trackBack	1249
6.222.3	Friends And Related Function Documentation	1249
6.222.3.1	RemoveTransactionAction	1249
6.223	activemq::commands::ConsumerControl Class Reference	1250
6.223.1	Constructor & Destructor Documentation	1251
6.223.1.1	ConsumerControl	1251
6.223.1.2	ConsumerControl	1251
6.223.1.3	~ConsumerControl	1251
6.223.2	Member Function Documentation	1251
6.223.2.1	cloneDataStructure	1251
6.223.2.2	copyDataStructure	1251
6.223.2.3	equals	1252
6.223.2.4	getConsumerId	1252
6.223.2.5	getConsumerId	1252
6.223.2.6	getDataStructureType	1252
6.223.2.7	getPrefetch	1253
6.223.2.8	isClose	1253

6.223.2.9 isFlush	1253
6.223.2.10 isStart	1253
6.223.2.11 isStop	1253
6.223.2.12 operator=	1253
6.223.2.13 setClose	1253
6.223.2.14 setConsumerId	1253
6.223.2.15 setFlush	1253
6.223.2.16 setPrefetch	1253
6.223.2.17 setStart	1253
6.223.2.18 setStop	1253
6.223.2.19 toString	1253
6.223.2.20 visit	1254
6.223.3 Field Documentation	1254
6.223.3.1 close	1254
6.223.3.2 consumerId	1254
6.223.3.3 flush	1254
6.223.3.4 ID_CONSUMERCONTROL	1254
6.223.3.5 prefetch	1254
6.223.3.6 start	1254
6.223.3.7 stop	1254
6.224activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class	
Reference	1255
6.224.1 Detailed Description	1255
6.224.2 Constructor & Destructor Documentation	1256
6.224.2.1 ConsumerControlMarshaller	1256
6.224.2.2 ~ConsumerControlMarshaller	1256
6.224.3 Member Function Documentation	1256
6.224.3.1 createObject	1256
6.224.3.2 getDataStructureType	1256
6.224.3.3 looseMarshal	1256
6.224.3.4 looseUnmarshal	1257
6.224.3.5 tightMarshal1	1257
6.224.3.6 tightMarshal2	1257
6.224.3.7 tightUnmarshal	1258
6.225activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller Class	
Reference	1259
6.225.1 Detailed Description	1259

6.225.2 Constructor & Destructor Documentation	1260
6.225.2.1 ConsumerControlMarshaller	1260
6.225.2.2 ~ConsumerControlMarshaller	1260
6.225.3 Member Function Documentation	1260
6.225.3.1 createObject	1260
6.225.3.2 getDataStructureType	1260
6.225.3.3 looseMarshal	1260
6.225.3.4 looseUnmarshal	1261
6.225.3.5 tightMarshal1	1261
6.225.3.6 tightMarshal2	1261
6.225.3.7 tightUnmarshal	1262
6.226activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller Class	
Reference	1263
6.226.1 Detailed Description	1263
6.226.2 Constructor & Destructor Documentation	1264
6.226.2.1 ConsumerControlMarshaller	1264
6.226.2.2 ~ConsumerControlMarshaller	1264
6.226.3 Member Function Documentation	1264
6.226.3.1 createObject	1264
6.226.3.2 getDataStructureType	1264
6.226.3.3 looseMarshal	1264
6.226.3.4 looseUnmarshal	1265
6.226.3.5 tightMarshal1	1265
6.226.3.6 tightMarshal2	1265
6.226.3.7 tightUnmarshal	1266
6.227activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller Class	
Reference	1267
6.227.1 Detailed Description	1267
6.227.2 Constructor & Destructor Documentation	1268
6.227.2.1 ConsumerControlMarshaller	1268
6.227.2.2 ~ConsumerControlMarshaller	1268
6.227.3 Member Function Documentation	1268
6.227.3.1 createObject	1268
6.227.3.2 getDataStructureType	1268
6.227.3.3 looseMarshal	1268
6.227.3.4 looseUnmarshal	1269
6.227.3.5 tightMarshal1	1269

6.227.3.6 tightMarshal2	1269
6.227.3.7 tightUnmarshal	1270
6.228activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller Class	
Reference	1271
6.228.1 Detailed Description	1271
6.228.2 Constructor & Destructor Documentation	1272
6.228.2.1 ConsumerControlMarshaller	1272
6.228.2.2 ~ConsumerControlMarshaller	1272
6.228.3 Member Function Documentation	1272
6.228.3.1 createObject	1272
6.228.3.2 getDataStructureType	1272
6.228.3.3 looseMarshal	1272
6.228.3.4 looseUnmarshal	1273
6.228.3.5 tightMarshal1	1273
6.228.3.6 tightMarshal2	1273
6.228.3.7 tightUnmarshal	1274
6.229activemq::commands::ConsumerId Class Reference	1275
6.229.1 Member Typedef Documentation	1276
6.229.1.1 COMPARATOR	1276
6.229.2 Constructor & Destructor Documentation	1276
6.229.2.1 ConsumerId	1276
6.229.2.2 ConsumerId	1276
6.229.2.3 ConsumerId	1276
6.229.2.4 ~ConsumerId	1276
6.229.3 Member Function Documentation	1276
6.229.3.1 cloneDataStructure	1276
6.229.3.2 compareTo	1276
6.229.3.3 copyDataStructure	1276
6.229.3.4 equals	1277
6.229.3.5 equals	1277
6.229.3.6 getConnectionId	1277
6.229.3.7 getConnectionId	1277
6.229.3.8 getDataStructureType	1277
6.229.3.9 getParentId	1278
6.229.3.10getSessionId	1278
6.229.3.11getValue	1278
6.229.3.12operator<	1278

6.229.3.13	operator=	1278
6.229.3.14	operator==	1278
6.229.3.15	setConnectionId	1278
6.229.3.16	setSessionId	1278
6.229.3.17	setValue	1278
6.229.3.18	toString	1278
6.229.4	Field Documentation	1278
6.229.4.1	connectionId	1278
6.229.4.2	ID_CONSUMERID	1278
6.229.4.3	sessionId	1279
6.229.4.4	value	1279
6.230	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference	1280
6.230.1	Detailed Description	1280
6.230.2	Constructor & Destructor Documentation	1281
6.230.2.1	ConsumerIdMarshaller	1281
6.230.2.2	~ConsumerIdMarshaller	1281
6.230.3	Member Function Documentation	1281
6.230.3.1	createObject	1281
6.230.3.2	getDataStructureType	1281
6.230.3.3	looseMarshal	1281
6.230.3.4	looseUnmarshal	1282
6.230.3.5	tightMarshal1	1282
6.230.3.6	tightMarshal2	1282
6.230.3.7	tightUnmarshal	1283
6.231	activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class Reference	1284
6.231.1	Detailed Description	1284
6.231.2	Constructor & Destructor Documentation	1285
6.231.2.1	ConsumerIdMarshaller	1285
6.231.2.2	~ConsumerIdMarshaller	1285
6.231.3	Member Function Documentation	1285
6.231.3.1	createObject	1285
6.231.3.2	getDataStructureType	1285
6.231.3.3	looseMarshal	1285
6.231.3.4	looseUnmarshal	1286
6.231.3.5	tightMarshal1	1286
6.231.3.6	tightMarshal2	1286

6.231.3.7 tightUnmarshal	1287
6.232activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference	1288
6.232.1 Detailed Description	1288
6.232.2 Constructor & Destructor Documentation	1289
6.232.2.1 ConsumerIdMarshaller	1289
6.232.2.2 ~ConsumerIdMarshaller	1289
6.232.3 Member Function Documentation	1289
6.232.3.1 createObject	1289
6.232.3.2 getDataStructureType	1289
6.232.3.3 looseMarshal	1289
6.232.3.4 looseUnmarshal	1290
6.232.3.5 tightMarshal1	1290
6.232.3.6 tightMarshal2	1290
6.232.3.7 tightUnmarshal	1291
6.233activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference	1292
6.233.1 Detailed Description	1292
6.233.2 Constructor & Destructor Documentation	1293
6.233.2.1 ConsumerIdMarshaller	1293
6.233.2.2 ~ConsumerIdMarshaller	1293
6.233.3 Member Function Documentation	1293
6.233.3.1 createObject	1293
6.233.3.2 getDataStructureType	1293
6.233.3.3 looseMarshal	1293
6.233.3.4 looseUnmarshal	1294
6.233.3.5 tightMarshal1	1294
6.233.3.6 tightMarshal2	1294
6.233.3.7 tightUnmarshal	1295
6.234activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference	1296
6.234.1 Detailed Description	1296
6.234.2 Constructor & Destructor Documentation	1297
6.234.2.1 ConsumerIdMarshaller	1297
6.234.2.2 ~ConsumerIdMarshaller	1297
6.234.3 Member Function Documentation	1297
6.234.3.1 createObject	1297
6.234.3.2 getDataStructureType	1297
6.234.3.3 looseMarshal	1297

6.234.3.4 looseUnmarshal	1298
6.234.3.5 tightMarshal1	1298
6.234.3.6 tightMarshal2	1298
6.234.3.7 tightUnmarshal	1299
6.235activemq::commands::ConsumerInfo Class Reference	1300
6.235.1 Constructor & Destructor Documentation	1302
6.235.1.1 ConsumerInfo	1302
6.235.1.2 ConsumerInfo	1302
6.235.1.3 ~ConsumerInfo	1302
6.235.2 Member Function Documentation	1302
6.235.2.1 cloneDataStructure	1302
6.235.2.2 copyDataStructure	1302
6.235.2.3 equals	1303
6.235.2.4 getAdditionalPredicate	1303
6.235.2.5 getAdditionalPredicate	1303
6.235.2.6 getBrokerPath	1303
6.235.2.7 getBrokerPath	1303
6.235.2.8 getConsumerId	1303
6.235.2.9 getConsumerId	1303
6.235.2.10getDataStructureType	1303
6.235.2.11getDestination	1304
6.235.2.12getDestination	1304
6.235.2.13getMaximumPendingMessageLimit	1304
6.235.2.14getNetworkConsumerPath	1304
6.235.2.15getNetworkConsumerPath	1304
6.235.2.16getPrefetchSize	1304
6.235.2.17getPriority	1304
6.235.2.18getSelector	1304
6.235.2.19getSelector	1304
6.235.2.20getSubscriptionName	1304
6.235.2.21getSubscriptionName	1304
6.235.2.22sBrowser	1304
6.235.2.23sConsumerInfo	1304
6.235.2.24sDispatchAsync	1305
6.235.2.25sExclusive	1305
6.235.2.26sNetworkSubscription	1305

6.235.2.27	isNoLocal	1305
6.235.2.28	isNoRangeAcks	1305
6.235.2.29	isOptimizedAcknowledge	1305
6.235.2.30	isRetroactive	1305
6.235.2.31	operator=	1305
6.235.2.32	setAdditionalPredicate	1305
6.235.2.33	setBrokerPath	1305
6.235.2.34	setBrowser	1305
6.235.2.35	setConsumerId	1305
6.235.2.36	setDestination	1305
6.235.2.37	setDispatchAsync	1305
6.235.2.38	setExclusive	1305
6.235.2.39	setMaximumPendingMessageLimit	1305
6.235.2.40	setNetworkConsumerPath	1305
6.235.2.41	setNetworkSubscription	1305
6.235.2.42	setNoLocal	1305
6.235.2.43	setNoRangeAcks	1305
6.235.2.44	setOptimizedAcknowledge	1305
6.235.2.45	setPrefetchSize	1305
6.235.2.46	setPriority	1305
6.235.2.47	setRetroactive	1305
6.235.2.48	setSelector	1305
6.235.2.49	setSubscriptionName	1305
6.235.2.50	toString	1305
6.235.2.51	visit	1306
6.235.3	Field Documentation	1307
6.235.3.1	additionalPredicate	1307
6.235.3.2	brokerPath	1307
6.235.3.3	browser	1307
6.235.3.4	consumerId	1307
6.235.3.5	destination	1307
6.235.3.6	dispatchAsync	1307
6.235.3.7	exclusive	1307
6.235.3.8	ID_CONSUMERINFO	1307
6.235.3.9	maximumPendingMessageLimit	1307
6.235.3.10	networkConsumerPath	1307

6.235.3.1	networkSubscription	1307
6.235.3.12	noLocal	1307
6.235.3.13	noRangeAcks	1307
6.235.3.14	optimizedAcknowledge	1307
6.235.3.15	prefetchSize	1307
6.235.3.16	priority	1307
6.235.3.17	retroactive	1307
6.235.3.18	selector	1307
6.235.3.19	subscriptionName	1307
6.236	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference	1309
6.236.1	Detailed Description	1309
6.236.2	Constructor & Destructor Documentation	1310
6.236.2.1	ConsumerInfoMarshaller	1310
6.236.2.2	~ConsumerInfoMarshaller	1310
6.236.3	Member Function Documentation	1310
6.236.3.1	createObject	1310
6.236.3.2	getDataStructureType	1310
6.236.3.3	looseMarshal	1310
6.236.3.4	looseUnmarshal	1311
6.236.3.5	tightMarshal1	1311
6.236.3.6	tightMarshal2	1311
6.236.3.7	tightUnmarshal	1312
6.237	activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference	1313
6.237.1	Detailed Description	1313
6.237.2	Constructor & Destructor Documentation	1314
6.237.2.1	ConsumerInfoMarshaller	1314
6.237.2.2	~ConsumerInfoMarshaller	1314
6.237.3	Member Function Documentation	1314
6.237.3.1	createObject	1314
6.237.3.2	getDataStructureType	1314
6.237.3.3	looseMarshal	1314
6.237.3.4	looseUnmarshal	1315
6.237.3.5	tightMarshal1	1315
6.237.3.6	tightMarshal2	1315
6.237.3.7	tightUnmarshal	1316

6.238activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference	1317
6.238.1 Detailed Description	1317
6.238.2 Constructor & Destructor Documentation	1318
6.238.2.1 ConsumerInfoMarshaller	1318
6.238.2.2 ~ConsumerInfoMarshaller	1318
6.238.3 Member Function Documentation	1318
6.238.3.1 createObject	1318
6.238.3.2 getDataStructureType	1318
6.238.3.3 looseMarshal	1318
6.238.3.4 looseUnmarshal	1319
6.238.3.5 tightMarshal1	1319
6.238.3.6 tightMarshal2	1319
6.238.3.7 tightUnmarshal	1320
6.239activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference	1321
6.239.1 Detailed Description	1321
6.239.2 Constructor & Destructor Documentation	1322
6.239.2.1 ConsumerInfoMarshaller	1322
6.239.2.2 ~ConsumerInfoMarshaller	1322
6.239.3 Member Function Documentation	1322
6.239.3.1 createObject	1322
6.239.3.2 getDataStructureType	1322
6.239.3.3 looseMarshal	1322
6.239.3.4 looseUnmarshal	1323
6.239.3.5 tightMarshal1	1323
6.239.3.6 tightMarshal2	1323
6.239.3.7 tightUnmarshal	1324
6.240activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference	1325
6.240.1 Detailed Description	1325
6.240.2 Constructor & Destructor Documentation	1326
6.240.2.1 ConsumerInfoMarshaller	1326
6.240.2.2 ~ConsumerInfoMarshaller	1326
6.240.3 Member Function Documentation	1326
6.240.3.1 createObject	1326
6.240.3.2 getDataStructureType	1326

6.240.3.3 looseMarshal	1326
6.240.3.4 looseUnmarshal	1327
6.240.3.5 tightMarshal1	1327
6.240.3.6 tightMarshal2	1327
6.240.3.7 tightUnmarshal	1328
6.241activemq::state::ConsumerState Class Reference	1329
6.241.1 Constructor & Destructor Documentation	1329
6.241.1.1 ConsumerState	1329
6.241.1.2 ~ConsumerState	1329
6.241.2 Member Function Documentation	1329
6.241.2.1 getInfo	1329
6.241.2.2 toString	1329
6.242activemq::commands::ControlCommand Class Reference	1330
6.242.1 Constructor & Destructor Documentation	1331
6.242.1.1 ControlCommand	1331
6.242.1.2 ControlCommand	1331
6.242.1.3 ~ControlCommand	1331
6.242.2 Member Function Documentation	1331
6.242.2.1 cloneDataStructure	1331
6.242.2.2 copyDataStructure	1331
6.242.2.3 equals	1331
6.242.2.4 getCommand	1332
6.242.2.5 getCommand	1332
6.242.2.6 getDataStructureType	1332
6.242.2.7 operator=	1332
6.242.2.8 setCommand	1332
6.242.2.9 toString	1332
6.242.2.10visit	1332
6.242.3 Field Documentation	1333
6.242.3.1 command	1333
6.242.3.2 ID_CONTROLCOMMAND	1333
6.243activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller Class Reference	1334
6.243.1 Detailed Description	1334
6.243.2 Constructor & Destructor Documentation	1335
6.243.2.1 ControlCommandMarshaller	1335
6.243.2.2 ~ControlCommandMarshaller	1335

6.243.3 Member Function Documentation	1335
6.243.3.1 createObject	1335
6.243.3.2 getDataStructureType	1335
6.243.3.3 looseMarshal	1335
6.243.3.4 looseUnmarshal	1336
6.243.3.5 tightMarshal1	1336
6.243.3.6 tightMarshal2	1336
6.243.3.7 tightUnmarshal	1337
6.244activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class	
Reference	1338
6.244.1 Detailed Description	1338
6.244.2 Constructor & Destructor Documentation	1339
6.244.2.1 ControlCommandMarshaller	1339
6.244.2.2 ~ControlCommandMarshaller	1339
6.244.3 Member Function Documentation	1339
6.244.3.1 createObject	1339
6.244.3.2 getDataStructureType	1339
6.244.3.3 looseMarshal	1339
6.244.3.4 looseUnmarshal	1340
6.244.3.5 tightMarshal1	1340
6.244.3.6 tightMarshal2	1340
6.244.3.7 tightUnmarshal	1341
6.245activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class	
Reference	1342
6.245.1 Detailed Description	1342
6.245.2 Constructor & Destructor Documentation	1343
6.245.2.1 ControlCommandMarshaller	1343
6.245.2.2 ~ControlCommandMarshaller	1343
6.245.3 Member Function Documentation	1343
6.245.3.1 createObject	1343
6.245.3.2 getDataStructureType	1343
6.245.3.3 looseMarshal	1343
6.245.3.4 looseUnmarshal	1344
6.245.3.5 tightMarshal1	1344
6.245.3.6 tightMarshal2	1344
6.245.3.7 tightUnmarshal	1345

6.246	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	Class	
	Reference		1346
6.246.1	Detailed Description		1346
6.246.2	Constructor & Destructor Documentation		1347
6.246.2.1	ControlCommandMarshaller		1347
6.246.2.2	~ControlCommandMarshaller		1347
6.246.3	Member Function Documentation		1347
6.246.3.1	createObject		1347
6.246.3.2	getDataStructureType		1347
6.246.3.3	looseMarshal		1347
6.246.3.4	looseUnmarshal		1348
6.246.3.5	tightMarshal1		1348
6.246.3.6	tightMarshal2		1348
6.246.3.7	tightUnmarshal		1349
6.247	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	Class	
	Reference		1350
6.247.1	Detailed Description		1350
6.247.2	Constructor & Destructor Documentation		1351
6.247.2.1	ControlCommandMarshaller		1351
6.247.2.2	~ControlCommandMarshaller		1351
6.247.3	Member Function Documentation		1351
6.247.3.1	createObject		1351
6.247.3.2	getDataStructureType		1351
6.247.3.3	looseMarshal		1351
6.247.3.4	looseUnmarshal		1352
6.247.3.5	tightMarshal1		1352
6.247.3.6	tightMarshal2		1352
6.247.3.7	tightUnmarshal		1353
6.248	decaf::util::concurrent::CountDownLatch	Class Reference	1354
6.248.1	Constructor & Destructor Documentation		1354
6.248.1.1	CountDownLatch		1354
6.248.1.2	~CountDownLatch		1354
6.248.2	Member Function Documentation		1354
6.248.2.1	await		1354
6.248.2.2	await		1355
6.248.2.3	countDown		1355
6.248.2.4	getCount		1355

6.249	activemq::commands::DataArrayResponse Class Reference	1356
6.249.1	Constructor & Destructor Documentation	1357
6.249.1.1	DataArrayResponse	1357
6.249.1.2	DataArrayResponse	1357
6.249.1.3	~DataArrayResponse	1357
6.249.2	Member Function Documentation	1357
6.249.2.1	cloneDataStructure	1357
6.249.2.2	copyDataStructure	1357
6.249.2.3	equals	1357
6.249.2.4	getData	1358
6.249.2.5	getData	1358
6.249.2.6	getDataStructureType	1358
6.249.2.7	operator=	1358
6.249.2.8	setData	1358
6.249.2.9	toString	1358
6.249.3	Field Documentation	1358
6.249.3.1	data	1358
6.249.3.2	ID_DATAARRAYRESPONSE	1358
6.250	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference	1359
6.250.1	Detailed Description	1359
6.250.2	Constructor & Destructor Documentation	1360
6.250.2.1	DataArrayResponseMarshaller	1360
6.250.2.2	~DataArrayResponseMarshaller	1360
6.250.3	Member Function Documentation	1360
6.250.3.1	createObject	1360
6.250.3.2	getDataStructureType	1360
6.250.3.3	looseMarshal	1360
6.250.3.4	looseUnmarshal	1361
6.250.3.5	tightMarshal1	1361
6.250.3.6	tightMarshal2	1362
6.250.3.7	tightUnmarshal	1362
6.251	activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class Reference	1363
6.251.1	Detailed Description	1363
6.251.2	Constructor & Destructor Documentation	1364
6.251.2.1	DataArrayResponseMarshaller	1364

6.251.2.2 ~DataArrayResponseMarshaller	1364
6.251.3 Member Function Documentation	1364
6.251.3.1 createObject	1364
6.251.3.2 getDataStructureType	1364
6.251.3.3 looseMarshal	1364
6.251.3.4 looseUnmarshal	1365
6.251.3.5 tightMarshal1	1365
6.251.3.6 tightMarshal2	1366
6.251.3.7 tightUnmarshal	1366
6.252activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class	
Reference	1367
6.252.1 Detailed Description	1367
6.252.2 Constructor & Destructor Documentation	1368
6.252.2.1 DataArrayResponseMarshaller	1368
6.252.2.2 ~DataArrayResponseMarshaller	1368
6.252.3 Member Function Documentation	1368
6.252.3.1 createObject	1368
6.252.3.2 getDataStructureType	1368
6.252.3.3 looseMarshal	1368
6.252.3.4 looseUnmarshal	1369
6.252.3.5 tightMarshal1	1369
6.252.3.6 tightMarshal2	1370
6.252.3.7 tightUnmarshal	1370
6.253activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class	
Reference	1371
6.253.1 Detailed Description	1371
6.253.2 Constructor & Destructor Documentation	1372
6.253.2.1 DataArrayResponseMarshaller	1372
6.253.2.2 ~DataArrayResponseMarshaller	1372
6.253.3 Member Function Documentation	1372
6.253.3.1 createObject	1372
6.253.3.2 getDataStructureType	1372
6.253.3.3 looseMarshal	1372
6.253.3.4 looseUnmarshal	1373
6.253.3.5 tightMarshal1	1373
6.253.3.6 tightMarshal2	1374
6.253.3.7 tightUnmarshal	1374

6.254	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class Reference	1375
6.254.1	Detailed Description	1375
6.254.2	Constructor & Destructor Documentation	1376
6.254.2.1	DataArrayResponseMarshaller	1376
6.254.2.2	~DataArrayResponseMarshaller	1376
6.254.3	Member Function Documentation	1376
6.254.3.1	createObject	1376
6.254.3.2	getDataStructureType	1376
6.254.3.3	looseMarshal	1376
6.254.3.4	looseUnmarshal	1377
6.254.3.5	tightMarshal1	1377
6.254.3.6	tightMarshal2	1378
6.254.3.7	tightUnmarshal	1378
6.255	decaf::io::DataInputStream Class Reference	1379
6.255.1	Detailed Description	1380
6.255.2	Constructor & Destructor Documentation	1380
6.255.2.1	DataInputStream	1380
6.255.2.2	~DataInputStream	1381
6.255.3	Member Function Documentation	1381
6.255.3.1	read	1381
6.255.3.2	read	1382
6.255.3.3	read	1382
6.255.3.4	readBoolean	1382
6.255.3.5	readByte	1383
6.255.3.6	readChar	1383
6.255.3.7	readDouble	1383
6.255.3.8	readFloat	1384
6.255.3.9	readFully	1384
6.255.3.10	readFully	1384
6.255.3.11	readInt	1385
6.255.3.12	readLong	1385
6.255.3.13	readShort	1386
6.255.3.14	readString	1386
6.255.3.15	readUnsignedByte	1386
6.255.3.16	readUnsignedShort	1386
6.255.3.17	readUTF	1387

6.255.3.18kip	1387
6.256decaf::io::DataOutputStream Class Reference	1388
6.256.1 Detailed Description	1389
6.256.2 Constructor & Destructor Documentation	1389
6.256.2.1 DataOutputStream	1389
6.256.2.2 ~DataOutputStream	1390
6.256.3 Member Function Documentation	1390
6.256.3.1 size	1390
6.256.3.2 write	1390
6.256.3.3 write	1390
6.256.3.4 write	1390
6.256.3.5 writeBoolean	1391
6.256.3.6 writeByte	1391
6.256.3.7 writeBytes	1391
6.256.3.8 writeChar	1392
6.256.3.9 writeChars	1392
6.256.3.10writeDouble	1392
6.256.3.11writeFloat	1392
6.256.3.12writeInt	1393
6.256.3.13writeLong	1393
6.256.3.14writeShort	1393
6.256.3.15writeUnsignedShort	1394
6.256.3.16writeUTF	1394
6.256.4 Field Documentation	1394
6.256.4.1 buffer	1394
6.256.4.2 written	1394
6.257activemq::commands::DataResponse Class Reference	1395
6.257.1 Constructor & Destructor Documentation	1396
6.257.1.1 DataResponse	1396
6.257.1.2 DataResponse	1396
6.257.1.3 ~DataResponse	1396
6.257.2 Member Function Documentation	1396
6.257.2.1 cloneDataStructure	1396
6.257.2.2 copyDataStructure	1396
6.257.2.3 equals	1396
6.257.2.4 getData	1397

6.257.2.5	getData	1397
6.257.2.6	getDataStructureType	1397
6.257.2.7	operator=	1397
6.257.2.8	setData	1397
6.257.2.9	toString	1397
6.257.3	Field Documentation	1397
6.257.3.1	data	1397
6.257.3.2	ID_DATARESPONSE	1397
6.258	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference	1398
6.258.1	Detailed Description	1398
6.258.2	Constructor & Destructor Documentation	1399
6.258.2.1	DataResponseMarshaller	1399
6.258.2.2	~DataResponseMarshaller	1399
6.258.3	Member Function Documentation	1399
6.258.3.1	createObject	1399
6.258.3.2	getDataStructureType	1399
6.258.3.3	looseMarshal	1399
6.258.3.4	looseUnmarshal	1400
6.258.3.5	tightMarshal1	1400
6.258.3.6	tightMarshal2	1401
6.258.3.7	tightUnmarshal	1401
6.259	activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference	1402
6.259.1	Detailed Description	1402
6.259.2	Constructor & Destructor Documentation	1403
6.259.2.1	DataResponseMarshaller	1403
6.259.2.2	~DataResponseMarshaller	1403
6.259.3	Member Function Documentation	1403
6.259.3.1	createObject	1403
6.259.3.2	getDataStructureType	1403
6.259.3.3	looseMarshal	1403
6.259.3.4	looseUnmarshal	1404
6.259.3.5	tightMarshal1	1404
6.259.3.6	tightMarshal2	1405
6.259.3.7	tightUnmarshal	1405

6.260activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference	1406
6.260.1 Detailed Description	1406
6.260.2 Constructor & Destructor Documentation	1407
6.260.2.1 DataResponseMarshaller	1407
6.260.2.2 ~DataResponseMarshaller	1407
6.260.3 Member Function Documentation	1407
6.260.3.1 createObject	1407
6.260.3.2 getDataStructureType	1407
6.260.3.3 looseMarshal	1407
6.260.3.4 looseUnmarshal	1408
6.260.3.5 tightMarshal1	1408
6.260.3.6 tightMarshal2	1409
6.260.3.7 tightUnmarshal	1409
6.261activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference	1410
6.261.1 Detailed Description	1410
6.261.2 Constructor & Destructor Documentation	1411
6.261.2.1 DataResponseMarshaller	1411
6.261.2.2 ~DataResponseMarshaller	1411
6.261.3 Member Function Documentation	1411
6.261.3.1 createObject	1411
6.261.3.2 getDataStructureType	1411
6.261.3.3 looseMarshal	1411
6.261.3.4 looseUnmarshal	1412
6.261.3.5 tightMarshal1	1412
6.261.3.6 tightMarshal2	1413
6.261.3.7 tightUnmarshal	1413
6.262activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference	1414
6.262.1 Detailed Description	1414
6.262.2 Constructor & Destructor Documentation	1415
6.262.2.1 DataResponseMarshaller	1415
6.262.2.2 ~DataResponseMarshaller	1415
6.262.3 Member Function Documentation	1415
6.262.3.1 createObject	1415
6.262.3.2 getDataStructureType	1415

6.262.3.3 looseMarshal	1415
6.262.3.4 looseUnmarshal	1416
6.262.3.5 tightMarshal1	1416
6.262.3.6 tightMarshal2	1417
6.262.3.7 tightUnmarshal	1417
6.263activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference	1418
6.263.1 Detailed Description	1418
6.263.2 Constructor & Destructor Documentation	1419
6.263.2.1 ~DataStreamMarshaller	1419
6.263.3 Member Function Documentation	1419
6.263.3.1 createObject	1419
6.263.3.2 getDataStructureType	1424
6.263.3.3 looseMarshal	1430
6.263.3.4 looseUnmarshal	1436
6.263.3.5 tightMarshal1	1442
6.263.3.6 tightMarshal2	1448
6.263.3.7 tightUnmarshal	1454
6.264activemq::commands::DataStructure Class Reference	1461
6.264.1 Constructor & Destructor Documentation	1461
6.264.1.1 ~DataStructure	1461
6.264.2 Member Function Documentation	1461
6.264.2.1 cloneDataStructure	1461
6.264.2.2 copyDataStructure	1462
6.264.2.3 equals	1463
6.264.2.4 getDataStructureType	1464
6.264.2.5 toString	1465
6.265decaf::util::Date Class Reference	1467
6.265.1 Detailed Description	1468
6.265.2 Constructor & Destructor Documentation	1468
6.265.2.1 Date	1468
6.265.2.2 Date	1468
6.265.2.3 Date	1468
6.265.2.4 ~Date	1468
6.265.3 Member Function Documentation	1468
6.265.3.1 after	1468
6.265.3.2 before	1468

6.265.3.3	compareTo	1469
6.265.3.4	equals	1469
6.265.3.5	getTime	1469
6.265.3.6	operator<	1469
6.265.3.7	operator=	1470
6.265.3.8	operator==	1470
6.265.3.9	setTime	1470
6.265.3.10	toString	1470
6.266	decaf::internal::DecafRuntime Class Reference	1472
6.266.1	Detailed Description	1472
6.266.2	Constructor & Destructor Documentation	1472
6.266.2.1	DecafRuntime	1472
6.266.2.2	~DecafRuntime	1472
6.266.3	Member Function Documentation	1472
6.266.3.1	getGlobalPool	1472
6.267	activemq::threads::DedicatedTaskRunner Class Reference	1473
6.267.1	Constructor & Destructor Documentation	1473
6.267.1.1	DedicatedTaskRunner	1473
6.267.1.2	~DedicatedTaskRunner	1473
6.267.2	Member Function Documentation	1473
6.267.2.1	run	1473
6.267.2.2	shutdown	1474
6.267.2.3	shutdown	1474
6.267.2.4	wakeup	1474
6.268	activemq::transport::DefaultTransportListener Class Reference	1475
6.268.1	Constructor & Destructor Documentation	1475
6.268.1.1	~DefaultTransportListener	1475
6.268.2	Member Function Documentation	1475
6.268.2.1	onCommand	1475
6.268.2.2	onException	1475
6.268.2.3	transportInterrupted	1476
6.268.2.4	transportResumed	1476
6.269	decaf::util::concurrent::Delayed Class Reference	1477
6.269.1	Detailed Description	1477
6.269.2	Constructor & Destructor Documentation	1477
6.269.2.1	~Delayed	1477

6.269.3 Member Function Documentation	1477
6.269.3.1 getDelay	1477
6.270cms::DeliveryMode Class Reference	1478
6.270.1 Detailed Description	1478
6.270.2 Member Enumeration Documentation	1478
6.270.2.1 DELIVERY_MODE	1478
6.270.3 Constructor & Destructor Documentation	1479
6.270.3.1 ~DeliveryMode	1479
6.271cms::Destination Class Reference	1480
6.271.1 Detailed Description	1480
6.271.2 Member Enumeration Documentation	1480
6.271.2.1 DestinationType	1480
6.271.3 Constructor & Destructor Documentation	1481
6.271.3.1 ~Destination	1481
6.271.4 Member Function Documentation	1481
6.271.4.1 clone	1481
6.271.4.2 copy	1481
6.271.4.3 getCMSProperties	1481
6.271.4.4 getDestinationType	1481
6.272activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference . .	1483
6.272.1 Field Documentation	1483
6.272.1.1 ANY_CHILD	1483
6.272.1.2 ANY_DESCENDENT	1483
6.273activemq::commands::DestinationInfo Class Reference	1484
6.273.1 Constructor & Destructor Documentation	1485
6.273.1.1 DestinationInfo	1485
6.273.1.2 DestinationInfo	1485
6.273.1.3 ~DestinationInfo	1485
6.273.2 Member Function Documentation	1485
6.273.2.1 cloneDataStructure	1485
6.273.2.2 copyDataStructure	1485
6.273.2.3 equals	1486
6.273.2.4 getBrokerPath	1486
6.273.2.5 getBrokerPath	1486
6.273.2.6 getConnectionId	1486
6.273.2.7 getConnectionId	1486

6.273.2.8	getDataStructureType	1486
6.273.2.9	getDestination	1487
6.273.2.10	getDestination	1487
6.273.2.11	getOperationType	1487
6.273.2.12	getTimeout	1487
6.273.2.13	operator=	1487
6.273.2.14	setBrokerPath	1487
6.273.2.15	setConnectionId	1487
6.273.2.16	setDestination	1487
6.273.2.17	setOperationType	1487
6.273.2.18	setTimeout	1487
6.273.2.19	toString	1487
6.273.2.20	visit	1487
6.273.3	Field Documentation	1488
6.273.3.1	brokerPath	1488
6.273.3.2	connectionId	1488
6.273.3.3	destination	1488
6.273.3.4	ID_DESTINATIONINFO	1488
6.273.3.5	operationType	1488
6.273.3.6	timeout	1488
6.274	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference	1489
6.274.1	Detailed Description	1489
6.274.2	Constructor & Destructor Documentation	1490
6.274.2.1	DestinationInfoMarshaller	1490
6.274.2.2	~DestinationInfoMarshaller	1490
6.274.3	Member Function Documentation	1490
6.274.3.1	createObject	1490
6.274.3.2	getDataStructureType	1490
6.274.3.3	looseMarshal	1490
6.274.3.4	looseUnmarshal	1491
6.274.3.5	tightMarshal1	1491
6.274.3.6	tightMarshal2	1491
6.274.3.7	tightUnmarshal	1492
6.275	activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference	1493
6.275.1	Detailed Description	1493

6.275.2 Constructor & Destructor Documentation	1494
6.275.2.1 DestinationInfoMarshaller	1494
6.275.2.2 ~DestinationInfoMarshaller	1494
6.275.3 Member Function Documentation	1494
6.275.3.1 createObject	1494
6.275.3.2 getDataStructureType	1494
6.275.3.3 looseMarshal	1494
6.275.3.4 looseUnmarshal	1495
6.275.3.5 tightMarshal1	1495
6.275.3.6 tightMarshal2	1495
6.275.3.7 tightUnmarshal	1496
6.276activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference	1497
6.276.1 Detailed Description	1497
6.276.2 Constructor & Destructor Documentation	1498
6.276.2.1 DestinationInfoMarshaller	1498
6.276.2.2 ~DestinationInfoMarshaller	1498
6.276.3 Member Function Documentation	1498
6.276.3.1 createObject	1498
6.276.3.2 getDataStructureType	1498
6.276.3.3 looseMarshal	1498
6.276.3.4 looseUnmarshal	1499
6.276.3.5 tightMarshal1	1499
6.276.3.6 tightMarshal2	1499
6.276.3.7 tightUnmarshal	1500
6.277activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference	1501
6.277.1 Detailed Description	1501
6.277.2 Constructor & Destructor Documentation	1502
6.277.2.1 DestinationInfoMarshaller	1502
6.277.2.2 ~DestinationInfoMarshaller	1502
6.277.3 Member Function Documentation	1502
6.277.3.1 createObject	1502
6.277.3.2 getDataStructureType	1502
6.277.3.3 looseMarshal	1502
6.277.3.4 looseUnmarshal	1503
6.277.3.5 tightMarshal1	1503

6.277.3.6 tightMarshal2	1503
6.277.3.7 tightUnmarshal	1504
6.278activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference	1505
6.278.1 Detailed Description	1505
6.278.2 Constructor & Destructor Documentation	1506
6.278.2.1 DestinationInfoMarshaller	1506
6.278.2.2 ~DestinationInfoMarshaller	1506
6.278.3 Member Function Documentation	1506
6.278.3.1 createObject	1506
6.278.3.2 getDataStructureType	1506
6.278.3.3 looseMarshal	1506
6.278.3.4 looseUnmarshal	1507
6.278.3.5 tightMarshal1	1507
6.278.3.6 tightMarshal2	1507
6.278.3.7 tightUnmarshal	1508
6.279activemq::cmsutil::DestinationResolver Class Reference	1509
6.279.1 Detailed Description	1509
6.279.2 Constructor & Destructor Documentation	1509
6.279.2.1 ~DestinationResolver	1509
6.279.3 Member Function Documentation	1509
6.279.3.1 destroy	1509
6.279.3.2 init	1509
6.279.3.3 resolveDestinationName	1510
6.280activemq::commands::DiscoveryEvent Class Reference	1511
6.280.1 Constructor & Destructor Documentation	1512
6.280.1.1 DiscoveryEvent	1512
6.280.1.2 DiscoveryEvent	1512
6.280.1.3 ~DiscoveryEvent	1512
6.280.2 Member Function Documentation	1512
6.280.2.1 cloneDataStructure	1512
6.280.2.2 copyDataStructure	1512
6.280.2.3 equals	1512
6.280.2.4 getBrokerName	1513
6.280.2.5 getBrokerName	1513
6.280.2.6 getDataStructureType	1513
6.280.2.7 getServiceName	1513

6.280.2.8	getServiceName	1513
6.280.2.9	operator=	1513
6.280.2.10	setBrokerName	1513
6.280.2.11	setServiceName	1513
6.280.2.12	toString	1513
6.280.3	Field Documentation	1514
6.280.3.1	brokerName	1514
6.280.3.2	ID_DISCOVERYEVENT	1514
6.280.3.3	serviceName	1514
6.281	activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference	1515
6.281.1	Detailed Description	1515
6.281.2	Constructor & Destructor Documentation	1516
6.281.2.1	DiscoveryEventMarshaller	1516
6.281.2.2	~DiscoveryEventMarshaller	1516
6.281.3	Member Function Documentation	1516
6.281.3.1	createObject	1516
6.281.3.2	getDataStructureType	1516
6.281.3.3	looseMarshal	1516
6.281.3.4	looseUnmarshal	1517
6.281.3.5	tightMarshal1	1517
6.281.3.6	tightMarshal2	1517
6.281.3.7	tightUnmarshal	1518
6.282	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference	1519
6.282.1	Detailed Description	1519
6.282.2	Constructor & Destructor Documentation	1520
6.282.2.1	DiscoveryEventMarshaller	1520
6.282.2.2	~DiscoveryEventMarshaller	1520
6.282.3	Member Function Documentation	1520
6.282.3.1	createObject	1520
6.282.3.2	getDataStructureType	1520
6.282.3.3	looseMarshal	1520
6.282.3.4	looseUnmarshal	1521
6.282.3.5	tightMarshal1	1521
6.282.3.6	tightMarshal2	1521
6.282.3.7	tightUnmarshal	1522

6.283activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference	1523
6.283.1 Detailed Description	1523
6.283.2 Constructor & Destructor Documentation	1524
6.283.2.1 DiscoveryEventMarshaller	1524
6.283.2.2 ~DiscoveryEventMarshaller	1524
6.283.3 Member Function Documentation	1524
6.283.3.1 createObject	1524
6.283.3.2 getDataStructureType	1524
6.283.3.3 looseMarshal	1524
6.283.3.4 looseUnmarshal	1525
6.283.3.5 tightMarshal1	1525
6.283.3.6 tightMarshal2	1525
6.283.3.7 tightUnmarshal	1526
6.284activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference	1527
6.284.1 Detailed Description	1527
6.284.2 Constructor & Destructor Documentation	1528
6.284.2.1 DiscoveryEventMarshaller	1528
6.284.2.2 ~DiscoveryEventMarshaller	1528
6.284.3 Member Function Documentation	1528
6.284.3.1 createObject	1528
6.284.3.2 getDataStructureType	1528
6.284.3.3 looseMarshal	1528
6.284.3.4 looseUnmarshal	1529
6.284.3.5 tightMarshal1	1529
6.284.3.6 tightMarshal2	1529
6.284.3.7 tightUnmarshal	1530
6.285activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference	1531
6.285.1 Detailed Description	1531
6.285.2 Constructor & Destructor Documentation	1532
6.285.2.1 DiscoveryEventMarshaller	1532
6.285.2.2 ~DiscoveryEventMarshaller	1532
6.285.3 Member Function Documentation	1532
6.285.3.1 createObject	1532
6.285.3.2 getDataStructureType	1532

6.285.3.3 looseMarshal	1532
6.285.3.4 looseUnmarshal	1533
6.285.3.5 tightMarshal1	1533
6.285.3.6 tightMarshal2	1533
6.285.3.7 tightUnmarshal	1534
6.286activemq::core::DispatchData Class Reference	1535
6.286.1 Detailed Description	1535
6.286.2 Constructor & Destructor Documentation	1535
6.286.2.1 DispatchData	1535
6.286.2.2 DispatchData	1535
6.286.3 Member Function Documentation	1535
6.286.3.1 getConsumerId	1535
6.286.3.2 getMessage	1535
6.287activemq::core::Dispatcher Class Reference	1536
6.287.1 Detailed Description	1536
6.287.2 Constructor & Destructor Documentation	1536
6.287.2.1 ~Dispatcher	1536
6.287.3 Member Function Documentation	1536
6.287.3.1 dispatch	1536
6.288decaf::lang::Double Class Reference	1537
6.288.1 Constructor & Destructor Documentation	1539
6.288.1.1 Double	1539
6.288.1.2 Double	1539
6.288.1.3 ~Double	1539
6.288.2 Member Function Documentation	1539
6.288.2.1 byteValue	1539
6.288.2.2 compare	1539
6.288.2.3 compareTo	1540
6.288.2.4 compareTo	1540
6.288.2.5 doubleToLongBits	1540
6.288.2.6 doubleToRawLongBits	1541
6.288.2.7 doubleValue	1541
6.288.2.8 equals	1542
6.288.2.9 equals	1542
6.288.2.1float Value	1542
6.288.2.1 lint Value	1542

6.288.2.12sInfinite	1542
6.288.2.13sInfinite	1543
6.288.2.14sNaN	1543
6.288.2.15sNaN	1543
6.288.2.16ongBitsToDouble	1543
6.288.2.17ongValue	1543
6.288.2.18operator<	1544
6.288.2.19operator<	1544
6.288.2.20operator==	1544
6.288.2.21operator==	1544
6.288.2.22parseDouble	1545
6.288.2.23hortValue	1545
6.288.2.24oHexString	1545
6.288.2.25oString	1546
6.288.2.26oString	1546
6.288.2.27valueOf	1546
6.288.2.28valueOf	1547
6.288.3 Field Documentation	1547
6.288.3.1 MAX_VALUE	1547
6.288.3.2 MIN_VALUE	1547
6.288.3.3 NaN	1547
6.288.3.4 NEGATIVE_INFINITY	1547
6.288.3.5 POSITIVE_INFINITY	1547
6.288.3.6 SIZE	1547
6.289decaf::internal::nio::DoubleArrayBuffer Class Reference	1548
6.289.1 Constructor & Destructor Documentation	1549
6.289.1.1 DoubleArrayBuffer	1549
6.289.1.2 DoubleArrayBuffer	1549
6.289.1.3 DoubleArrayBuffer	1550
6.289.1.4 DoubleArrayBuffer	1550
6.289.1.5 ~DoubleArrayBuffer	1550
6.289.2 Member Function Documentation	1550
6.289.2.1 array	1550
6.289.2.2 arrayOffset	1551
6.289.2.3 asReadOnlyBuffer	1551
6.289.2.4 compact	1552

6.289.2.5 duplicate	1552
6.289.2.6 get	1552
6.289.2.7 get	1553
6.289.2.8 hasArray	1553
6.289.2.9 isReadOnly	1553
6.289.2.10put	1553
6.289.2.11put	1554
6.289.2.12setReadOnly	1554
6.289.2.13slice	1554
6.290decaf::nio::DoubleBuffer Class Reference	1556
6.290.1 Detailed Description	1558
6.290.2 Constructor & Destructor Documentation	1558
6.290.2.1 DoubleBuffer	1558
6.290.2.2 ~DoubleBuffer	1558
6.290.3 Member Function Documentation	1558
6.290.3.1 allocate	1558
6.290.3.2 array	1559
6.290.3.3 arrayOffset	1559
6.290.3.4 asReadOnlyBuffer	1559
6.290.3.5 compact	1560
6.290.3.6 compareTo	1560
6.290.3.7 duplicate	1560
6.290.3.8 equals	1561
6.290.3.9 get	1561
6.290.3.10get	1561
6.290.3.11get	1562
6.290.3.12get	1562
6.290.3.13hasArray	1562
6.290.3.14operator<	1562
6.290.3.15operator==	1563
6.290.3.16put	1563
6.290.3.17put	1563
6.290.3.18put	1564
6.290.3.19put	1564
6.290.3.20put	1565
6.290.3.21slice	1565

6.290.3.22	toString	1565
6.290.3.23	wrap	1566
6.290.3.24	wrap	1566
6.291	decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference	1567
6.292	activemq::cmsutil::DynamicDestinationResolver Class Reference	1568
6.292.1	Detailed Description	1568
6.292.2	Constructor & Destructor Documentation	1568
6.292.2.1	~DynamicDestinationResolver	1568
6.292.3	Member Function Documentation	1568
6.292.3.1	destroy	1568
6.292.3.2	init	1569
6.292.3.3	resolveDestinationName	1569
6.293	decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference	1570
6.293.1	Constructor & Destructor Documentation	1570
6.293.1.1	Entry	1570
6.293.1.2	~Entry	1570
6.293.2	Member Function Documentation	1570
6.293.2.1	getKey	1570
6.293.2.2	getValue	1570
6.293.2.3	setValue	1570
6.294	decaf::io::EOFException Class Reference	1571
6.294.1	Constructor & Destructor Documentation	1571
6.294.1.1	EOFException	1571
6.294.1.2	EOFException	1571
6.294.1.3	EOFException	1572
6.294.1.4	EOFException	1572
6.294.1.5	EOFException	1572
6.294.1.6	EOFException	1572
6.294.1.7	~EOFException	1573
6.294.2	Member Function Documentation	1573
6.294.2.1	clone	1573
6.295	decaf::lang::Exception Class Reference	1574
6.295.1	Constructor & Destructor Documentation	1575
6.295.1.1	Exception	1575
6.295.1.2	Exception	1576
6.295.1.3	Exception	1576

6.295.1.4 Exception	1576
6.295.1.5 Exception	1576
6.295.1.6 ~Exception	1577
6.295.2 Member Function Documentation	1577
6.295.2.1 buildMessage	1577
6.295.2.2 clone	1577
6.295.2.3 getCause	1577
6.295.2.4 getMessage	1578
6.295.2.5 getStackTrace	1578
6.295.2.6 getStackTraceString	1578
6.295.2.7 initCause	1578
6.295.2.8 operator=	1579
6.295.2.9 printStackTrace	1579
6.295.2.10 printStackTrace	1579
6.295.2.11 setMark	1579
6.295.2.12 setMessage	1579
6.295.2.13 setStackTrace	1580
6.295.2.14 what	1580
6.295.3 Field Documentation	1580
6.295.3.1 cause	1580
6.295.3.2 message	1580
6.295.3.3 stackTrace	1580
6.296 cms::ExceptionListener Class Reference	1581
6.296.1 Detailed Description	1581
6.296.2 Constructor & Destructor Documentation	1581
6.296.2.1 ~ExceptionListener	1581
6.296.3 Member Function Documentation	1581
6.296.3.1 onException	1581
6.297 activemq::commands::ExceptionResponse Class Reference	1582
6.297.1 Constructor & Destructor Documentation	1583
6.297.1.1 ExceptionResponse	1583
6.297.1.2 ExceptionResponse	1583
6.297.1.3 ~ExceptionResponse	1583
6.297.2 Member Function Documentation	1583
6.297.2.1 cloneDataStructure	1583
6.297.2.2 copyDataStructure	1583

6.297.2.3 equals	1583
6.297.2.4 getDataStructureType	1583
6.297.2.5 getException	1584
6.297.2.6 getException	1584
6.297.2.7 operator=	1584
6.297.2.8 setException	1584
6.297.2.9 toString	1584
6.297.3 Field Documentation	1584
6.297.3.1 exception	1584
6.297.3.2 ID_EXCEPTIONRESPONSE	1584
6.298activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class	
Reference	1585
6.298.1 Detailed Description	1585
6.298.2 Constructor & Destructor Documentation	1586
6.298.2.1 ExceptionResponseMarshaller	1586
6.298.2.2 ~ExceptionResponseMarshaller	1586
6.298.3 Member Function Documentation	1586
6.298.3.1 createObject	1586
6.298.3.2 getDataStructureType	1586
6.298.3.3 looseMarshal	1586
6.298.3.4 looseUnmarshal	1587
6.298.3.5 tightMarshal1	1587
6.298.3.6 tightMarshal2	1588
6.298.3.7 tightUnmarshal	1588
6.299activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class	
Reference	1589
6.299.1 Detailed Description	1589
6.299.2 Constructor & Destructor Documentation	1590
6.299.2.1 ExceptionResponseMarshaller	1590
6.299.2.2 ~ExceptionResponseMarshaller	1590
6.299.3 Member Function Documentation	1590
6.299.3.1 createObject	1590
6.299.3.2 getDataStructureType	1590
6.299.3.3 looseMarshal	1590
6.299.3.4 looseUnmarshal	1591
6.299.3.5 tightMarshal1	1591
6.299.3.6 tightMarshal2	1592

6.299.3.7 tightUnmarshal	1592
6.300activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class	
Reference	1593
6.300.1 Detailed Description	1593
6.300.2 Constructor & Destructor Documentation	1594
6.300.2.1 ExceptionResponseMarshaller	1594
6.300.2.2 ~ExceptionResponseMarshaller	1594
6.300.3 Member Function Documentation	1594
6.300.3.1 createObject	1594
6.300.3.2 getDataStructureType	1594
6.300.3.3 looseMarshal	1594
6.300.3.4 looseUnmarshal	1595
6.300.3.5 tightMarshal1	1595
6.300.3.6 tightMarshal2	1596
6.300.3.7 tightUnmarshal	1596
6.301activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class	
Reference	1597
6.301.1 Detailed Description	1597
6.301.2 Constructor & Destructor Documentation	1598
6.301.2.1 ExceptionResponseMarshaller	1598
6.301.2.2 ~ExceptionResponseMarshaller	1598
6.301.3 Member Function Documentation	1598
6.301.3.1 createObject	1598
6.301.3.2 getDataStructureType	1598
6.301.3.3 looseMarshal	1598
6.301.3.4 looseUnmarshal	1599
6.301.3.5 tightMarshal1	1599
6.301.3.6 tightMarshal2	1600
6.301.3.7 tightUnmarshal	1600
6.302activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class	
Reference	1601
6.302.1 Detailed Description	1601
6.302.2 Constructor & Destructor Documentation	1602
6.302.2.1 ExceptionResponseMarshaller	1602
6.302.2.2 ~ExceptionResponseMarshaller	1602
6.302.3 Member Function Documentation	1602
6.302.3.1 createObject	1602

6.302.3.2	getDataStructureType	1602
6.302.3.3	looseMarshal	1602
6.302.3.4	looseUnmarshal	1603
6.302.3.5	tightMarshal1	1603
6.302.3.6	tightMarshal2	1604
6.302.3.7	tightUnmarshal	1604
6.303	decaf::util::concurrent::ExecutionException Class Reference	1605
6.303.1	Constructor & Destructor Documentation	1605
6.303.1.1	ExecutionException	1605
6.303.1.2	ExecutionException	1605
6.303.1.3	ExecutionException	1606
6.303.1.4	ExecutionException	1606
6.303.1.5	ExecutionException	1606
6.303.1.6	ExecutionException	1606
6.303.1.7	~ExecutionException	1607
6.303.2	Member Function Documentation	1607
6.303.2.1	clone	1607
6.304	decaf::util::concurrent::Executor Class Reference	1608
6.304.1	Detailed Description	1608
6.304.2	Constructor & Destructor Documentation	1609
6.304.2.1	~Executor	1609
6.304.3	Member Function Documentation	1609
6.304.3.1	execute	1609
6.305	decaf::util::concurrent::ExecutorService Class Reference	1610
6.305.1	Detailed Description	1610
6.305.2	Constructor & Destructor Documentation	1611
6.305.2.1	~ExecutorService	1611
6.305.3	Member Function Documentation	1611
6.305.3.1	awaitTermination	1611
6.306	activemq::transport::failover::FailoverTransport Class Reference	1612
6.306.1	Constructor & Destructor Documentation	1614
6.306.1.1	FailoverTransport	1614
6.306.1.2	~FailoverTransport	1614
6.306.2	Member Function Documentation	1614
6.306.2.1	add	1614
6.306.2.2	addURI	1615

6.306.2.3 close	1615
6.306.2.4 getBackOffMultiplier	1616
6.306.2.5 getBackupPoolSize	1616
6.306.2.6 getInitialReconnectDelay	1616
6.306.2.7 getMaxCacheSize	1616
6.306.2.8 getMaxReconnectAttempts	1616
6.306.2.9 getMaxReconnectDelay	1616
6.306.2.10getReconnectDelay	1616
6.306.2.11getRemoteAddress	1616
6.306.2.12getTimeout	1616
6.306.2.13getTransportListener	1616
6.306.2.14handleTransportFailure	1617
6.306.2.15sBackup	1617
6.306.2.16sClosed	1617
6.306.2.17sConnected	1617
6.306.2.18sFault Tolerant	1617
6.306.2.19sInitialized	1618
6.306.2.20sPending	1618
6.306.2.21sRandomize	1618
6.306.2.22sTrackMessages	1618
6.306.2.23sUseExponentialBackOff	1618
6.306.2.24terate	1618
6.306.2.25narrow	1618
6.306.2.26neway	1619
6.306.2.27reconnect	1619
6.306.2.28reconnect	1619
6.306.2.29removeURI	1619
6.306.2.30request	1620
6.306.2.31request	1620
6.306.2.32restoreTransport	1620
6.306.2.33etBackOffMultiplier	1621
6.306.2.34etBackup	1621
6.306.2.35etBackupPoolSize	1621
6.306.2.36etInitialized	1621
6.306.2.37etInitialReconnectDelay	1622
6.306.2.38etMaxCacheSize	1622

6.306.2.39	setMaxReconnectAttempts	1622
6.306.2.40	setMaxReconnectDelay	1622
6.306.2.41	setRandomize	1622
6.306.2.42	setReconnectDelay	1622
6.306.2.43	setTimeout	1622
6.306.2.44	setTrackMessages	1622
6.306.2.45	setTransportListener	1622
6.306.2.46	setUseExponentialBackOff	1622
6.306.2.47	setWireFormat	1622
6.306.2.48	start	1623
6.306.2.49	stop	1623
6.306.3	Friends And Related Function Documentation	1623
6.306.3.1	FailoverTransportListener	1623
6.307	activemq::transport::failover::FailoverTransportFactory Class Reference	1624
6.307.1	Detailed Description	1624
6.307.2	Constructor & Destructor Documentation	1625
6.307.2.1	~FailoverTransportFactory	1625
6.307.3	Member Function Documentation	1625
6.307.3.1	create	1625
6.307.3.2	createComposite	1625
6.307.3.3	doCreateComposite	1625
6.308	activemq::transport::failover::FailoverTransportListener Class Reference	1627
6.308.1	Detailed Description	1627
6.308.2	Constructor & Destructor Documentation	1628
6.308.2.1	FailoverTransportListener	1628
6.308.2.2	~FailoverTransportListener	1628
6.308.3	Member Function Documentation	1628
6.308.3.1	onCommand	1628
6.308.3.2	onException	1628
6.308.3.3	transportInterrupted	1628
6.308.3.4	transportResumed	1628
6.309	decaf::util::logging::Filter Class Reference	1630
6.309.1	Detailed Description	1630
6.309.2	Constructor & Destructor Documentation	1630
6.309.2.1	~Filter	1630
6.309.3	Member Function Documentation	1630

6.309.3.1 isLoggable	1630
6.310decaf::io::FilterInputStream Class Reference	1631
6.310.1 Detailed Description	1632
6.310.2 Constructor & Destructor Documentation	1633
6.310.2.1 FilterInputStream	1633
6.310.2.2 ~FilterInputStream	1633
6.310.3 Member Function Documentation	1633
6.310.3.1 available	1633
6.310.3.2 close	1633
6.310.3.3 isClosed	1634
6.310.3.4 lock	1634
6.310.3.5 mark	1634
6.310.3.6 markSupported	1634
6.310.3.7 notify	1635
6.310.3.8 notifyAll	1635
6.310.3.9 read	1635
6.310.3.10 read	1636
6.310.3.11 reset	1636
6.310.3.12 skip	1637
6.310.3.13 tryLock	1637
6.310.3.14 unlock	1637
6.310.3.15 wait	1638
6.310.3.16 wait	1638
6.310.3.17 wait	1639
6.310.4 Field Documentation	1639
6.310.4.1 closed	1639
6.310.4.2 inputStream	1639
6.310.4.3 mutex	1639
6.310.4.4 own	1639
6.311decaf::io::FilterOutputStream Class Reference	1640
6.311.1 Detailed Description	1641
6.311.2 Constructor & Destructor Documentation	1641
6.311.2.1 FilterOutputStream	1641
6.311.2.2 ~FilterOutputStream	1642
6.311.3 Member Function Documentation	1642
6.311.3.1 close	1642

6.311.3.2 flush	1642
6.311.3.3 isClosed	1642
6.311.3.4 lock	1643
6.311.3.5 notify	1643
6.311.3.6 notifyAll	1643
6.311.3.7 tryLock	1643
6.311.3.8 unlock	1644
6.311.3.9 wait	1644
6.311.3.10wait	1644
6.311.3.11wait	1645
6.311.3.12write	1645
6.311.3.13write	1646
6.311.3.14write	1646
6.311.4 Field Documentation	1646
6.311.4.1 closed	1646
6.311.4.2 mutex	1646
6.311.4.3 outputStream	1646
6.311.4.4 own	1646
6.312decaf::lang::Float Class Reference	1647
6.312.1 Constructor & Destructor Documentation	1649
6.312.1.1 Float	1649
6.312.1.2 Float	1649
6.312.1.3 Float	1649
6.312.1.4 ~Float	1649
6.312.2 Member Function Documentation	1649
6.312.2.1 byteValue	1649
6.312.2.2 compare	1650
6.312.2.3 compareTo	1650
6.312.2.4 compareTo	1650
6.312.2.5 doubleValue	1650
6.312.2.6 equals	1651
6.312.2.7 equals	1651
6.312.2.8 floatToIntBits	1651
6.312.2.9 floatToRawIntBits	1652
6.312.2.10float Value	1652
6.312.2.11intBitsToFloat	1652

6.312.2.12	int Value	1653
6.312.2.13	sInfinite	1653
6.312.2.14	sInfinite	1653
6.312.2.15	sNaN	1653
6.312.2.16	sNaN	1653
6.312.2.17	long Value	1653
6.312.2.18	operator<	1654
6.312.2.19	operator<	1654
6.312.2.20	operator==	1654
6.312.2.21	operator==	1654
6.312.2.22	parseFloat	1655
6.312.2.23	short Value	1655
6.312.2.24	oHexString	1655
6.312.2.25	oString	1656
6.312.2.26	oString	1656
6.312.2.27	valueOf	1656
6.312.2.28	valueOf	1657
6.312.3	Field Documentation	1657
6.312.3.1	MAX_VALUE	1657
6.312.3.2	MIN_VALUE	1657
6.312.3.3	NaN	1657
6.312.3.4	NEGATIVE_INFINITY	1657
6.312.3.5	POSITIVE_INFINITY	1657
6.312.3.6	SIZE	1657
6.313	decaf::internal::nio::FloatArrayBuffer Class Reference	1658
6.313.1	Constructor & Destructor Documentation	1659
6.313.1.1	FloatArrayBuffer	1659
6.313.1.2	FloatArrayBuffer	1659
6.313.1.3	FloatArrayBuffer	1660
6.313.1.4	FloatArrayBuffer	1660
6.313.1.5	~FloatArrayBuffer	1660
6.313.2	Member Function Documentation	1660
6.313.2.1	array	1660
6.313.2.2	arrayOffset	1661
6.313.2.3	asReadOnlyBuffer	1661
6.313.2.4	compact	1661

6.313.2.5 duplicate	1662
6.313.2.6 get	1662
6.313.2.7 get	1662
6.313.2.8 hasArray	1663
6.313.2.9 isReadOnly	1663
6.313.2.10put	1663
6.313.2.11put	1664
6.313.2.12setReadOnly	1664
6.313.2.13slice	1664
6.314decaf::nio::FloatBuffer Class Reference	1665
6.314.1 Detailed Description	1667
6.314.2 Constructor & Destructor Documentation	1667
6.314.2.1 FloatBuffer	1667
6.314.2.2 ~FloatBuffer	1667
6.314.3 Member Function Documentation	1667
6.314.3.1 allocate	1667
6.314.3.2 array	1667
6.314.3.3 arrayOffset	1668
6.314.3.4 asReadOnlyBuffer	1668
6.314.3.5 compact	1668
6.314.3.6 compareTo	1669
6.314.3.7 duplicate	1669
6.314.3.8 equals	1669
6.314.3.9 get	1670
6.314.3.10get	1670
6.314.3.11get	1670
6.314.3.12get	1671
6.314.3.13hasArray	1671
6.314.3.14operator<	1671
6.314.3.15operator==	1672
6.314.3.16put	1672
6.314.3.17put	1672
6.314.3.18put	1673
6.314.3.19put	1673
6.314.3.20put	1673
6.314.3.21slice	1674

6.314.3.22	toString	1674
6.314.3.23	wrap	1674
6.314.3.24	wrap	1675
6.315	activemq::commands::FlushCommand Class Reference	1676
6.315.1	Constructor & Destructor Documentation	1677
6.315.1.1	FlushCommand	1677
6.315.1.2	FlushCommand	1677
6.315.1.3	~FlushCommand	1677
6.315.2	Member Function Documentation	1677
6.315.2.1	cloneDataStructure	1677
6.315.2.2	copyDataStructure	1677
6.315.2.3	equals	1677
6.315.2.4	getDataStructureType	1677
6.315.2.5	operator=	1678
6.315.2.6	toString	1678
6.315.2.7	visit	1678
6.315.3	Field Documentation	1678
6.315.3.1	ID_FLUSHCOMMAND	1678
6.316	activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference	1679
6.316.1	Detailed Description	1679
6.316.2	Constructor & Destructor Documentation	1680
6.316.2.1	FlushCommandMarshaller	1680
6.316.2.2	~FlushCommandMarshaller	1680
6.316.3	Member Function Documentation	1680
6.316.3.1	createObject	1680
6.316.3.2	getDataStructureType	1680
6.316.3.3	looseMarshal	1680
6.316.3.4	looseUnmarshal	1681
6.316.3.5	tightMarshal1	1681
6.316.3.6	tightMarshal2	1681
6.316.3.7	tightUnmarshal	1682
6.317	activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller Class Reference	1683
6.317.1	Detailed Description	1683
6.317.2	Constructor & Destructor Documentation	1684
6.317.2.1	FlushCommandMarshaller	1684

6.317.2.2	~FlushCommandMarshaller	1684
6.317.3	Member Function Documentation	1684
6.317.3.1	createObject	1684
6.317.3.2	getDataStructureType	1684
6.317.3.3	looseMarshal	1684
6.317.3.4	looseUnmarshal	1685
6.317.3.5	tightMarshal1	1685
6.317.3.6	tightMarshal2	1685
6.317.3.7	tightUnmarshal	1686
6.318	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference	1687
6.318.1	Detailed Description	1687
6.318.2	Constructor & Destructor Documentation	1688
6.318.2.1	FlushCommandMarshaller	1688
6.318.2.2	~FlushCommandMarshaller	1688
6.318.3	Member Function Documentation	1688
6.318.3.1	createObject	1688
6.318.3.2	getDataStructureType	1688
6.318.3.3	looseMarshal	1688
6.318.3.4	looseUnmarshal	1689
6.318.3.5	tightMarshal1	1689
6.318.3.6	tightMarshal2	1689
6.318.3.7	tightUnmarshal	1690
6.319	activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference	1691
6.319.1	Detailed Description	1691
6.319.2	Constructor & Destructor Documentation	1692
6.319.2.1	FlushCommandMarshaller	1692
6.319.2.2	~FlushCommandMarshaller	1692
6.319.3	Member Function Documentation	1692
6.319.3.1	createObject	1692
6.319.3.2	getDataStructureType	1692
6.319.3.3	looseMarshal	1692
6.319.3.4	looseUnmarshal	1693
6.319.3.5	tightMarshal1	1693
6.319.3.6	tightMarshal2	1693
6.319.3.7	tightUnmarshal	1694

6.320activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference	1695
6.320.1 Detailed Description	1695
6.320.2 Constructor & Destructor Documentation	1696
6.320.2.1 FlushCommandMarshaller	1696
6.320.2.2 ~FlushCommandMarshaller	1696
6.320.3 Member Function Documentation	1696
6.320.3.1 createObject	1696
6.320.3.2 getDataStructureType	1696
6.320.3.3 looseMarshal	1696
6.320.3.4 looseUnmarshal	1697
6.320.3.5 tightMarshal1	1697
6.320.3.6 tightMarshal2	1697
6.320.3.7 tightUnmarshal	1698
6.321decaf::util::logging::Formatter Class Reference	1699
6.321.1 Detailed Description	1699
6.321.2 Constructor & Destructor Documentation	1699
6.321.2.1 ~Formatter	1699
6.321.3 Member Function Documentation	1699
6.321.3.1 format	1699
6.321.3.2 formatMessage	1700
6.321.3.3 getHead	1700
6.321.3.4 getTail	1700
6.322decaf::util::concurrent::Future< V > Class Template Reference	1701
6.322.1 Detailed Description	1701
6.322.2 Constructor & Destructor Documentation	1702
6.322.2.1 ~Future	1702
6.322.3 Member Function Documentation	1702
6.322.3.1 cancel	1702
6.322.3.2 get	1702
6.322.3.3 get	1703
6.322.3.4 isCancelled	1703
6.322.3.5 isDone	1703
6.323activemq::transport::correlator::FutureResponse Class Reference	1704
6.323.1 Detailed Description	1704
6.323.2 Constructor & Destructor Documentation	1704
6.323.2.1 FutureResponse	1704

6.323.2.2 ~FutureResponse	1704
6.323.3 Member Function Documentation	1704
6.323.3.1 getResponse	1704
6.323.3.2 getResponse	1704
6.323.3.3 getResponse	1705
6.323.3.4 getResponse	1705
6.323.3.5 setResponse	1705
6.324decaf::security::GeneralSecurityException Class Reference	1706
6.324.1 Constructor & Destructor Documentation	1706
6.324.1.1 GeneralSecurityException	1706
6.324.1.2 GeneralSecurityException	1706
6.324.1.3 GeneralSecurityException	1707
6.324.1.4 GeneralSecurityException	1707
6.324.1.5 GeneralSecurityException	1707
6.324.1.6 GeneralSecurityException	1707
6.324.1.7 ~GeneralSecurityException	1708
6.324.2 Member Function Documentation	1708
6.324.2.1 clone	1708
6.325decaf::util::logging::Handler Class Reference	1709
6.325.1 Detailed Description	1709
6.325.2 Constructor & Destructor Documentation	1710
6.325.2.1 ~Handler	1710
6.325.3 Member Function Documentation	1710
6.325.3.1 flush	1710
6.325.3.2 getFilter	1710
6.325.3.3 getFormatter	1710
6.325.3.4 getLevel	1710
6.325.3.5 isLoggable	1710
6.325.3.6 publish	1711
6.325.3.7 setFilter	1711
6.325.3.8 setFormatter	1711
6.325.3.9 setLevel	1711
6.326decaf::internal::util::HexStringParser Class Reference	1713
6.326.1 Constructor & Destructor Documentation	1713
6.326.1.1 HexStringParser	1713
6.326.1.2 ~HexStringParser	1713

6.326.2 Member Function Documentation	1713
6.326.2.1 parse	1713
6.326.2.2 parseDouble	1714
6.326.2.3 parseFloat	1714
6.327activemq::wireformat::openwire::utils::HexTable Class Reference	1715
6.327.1 Detailed Description	1715
6.327.2 Constructor & Destructor Documentation	1715
6.327.2.1 HexTable	1715
6.327.2.2 ~HexTable	1715
6.327.3 Member Function Documentation	1715
6.327.3.1 operator[]	1715
6.327.3.2 operator[]	1715
6.327.3.3 size	1716
6.328decaf::net::HttpRetryException Class Reference	1717
6.328.1 Constructor & Destructor Documentation	1717
6.328.1.1 HttpRetryException	1717
6.328.1.2 HttpRetryException	1717
6.328.1.3 HttpRetryException	1718
6.328.1.4 HttpRetryException	1718
6.328.1.5 HttpRetryException	1718
6.328.1.6 HttpRetryException	1718
6.328.1.7 ~HttpRetryException	1719
6.328.2 Member Function Documentation	1719
6.328.2.1 clone	1719
6.329decaf::lang::exceptions::IllegalArgumentException Class Reference	1720
6.329.1 Constructor & Destructor Documentation	1720
6.329.1.1 IllegalArgumentException	1720
6.329.1.2 IllegalArgumentException	1720
6.329.1.3 IllegalArgumentException	1721
6.329.1.4 IllegalArgumentException	1721
6.329.1.5 IllegalArgumentException	1721
6.329.1.6 IllegalArgumentException	1721
6.329.1.7 ~IllegalArgumentException	1722
6.329.2 Member Function Documentation	1722
6.329.2.1 clone	1722
6.330decaf::lang::exceptions::IllegalMonitorStateException Class Reference	1723

6.330.1 Constructor & Destructor Documentation	1723
6.330.1.1 IllegalMonitorStateException	1723
6.330.1.2 IllegalMonitorStateException	1723
6.330.1.3 IllegalMonitorStateException	1724
6.330.1.4 IllegalMonitorStateException	1724
6.330.1.5 IllegalMonitorStateException	1724
6.330.1.6 IllegalMonitorStateException	1724
6.330.1.7 ~IllegalMonitorStateException	1725
6.330.2 Member Function Documentation	1725
6.330.2.1 clone	1725
6.331decaf::lang::exceptions::IllegalStateException Class Reference	1726
6.331.1 Constructor & Destructor Documentation	1726
6.331.1.1 IllegalStateException	1726
6.331.1.2 IllegalStateException	1726
6.331.1.3 IllegalStateException	1727
6.331.1.4 IllegalStateException	1727
6.331.1.5 IllegalStateException	1727
6.331.1.6 IllegalStateException	1727
6.331.1.7 ~IllegalStateException	1728
6.331.2 Member Function Documentation	1728
6.331.2.1 clone	1728
6.332cms::IllegalStateException Class Reference	1729
6.332.1 Detailed Description	1729
6.332.2 Constructor & Destructor Documentation	1729
6.332.2.1 IllegalStateException	1729
6.332.2.2 IllegalStateException	1729
6.332.2.3 IllegalStateException	1729
6.332.2.4 IllegalStateException	1729
6.332.2.5 ~IllegalStateException	1729
6.333decaf::lang::exceptions::IllegalThreadStateException Class Reference	1730
6.333.1 Constructor & Destructor Documentation	1730
6.333.1.1 IllegalThreadStateException	1730
6.333.1.2 IllegalThreadStateException	1730
6.333.1.3 IllegalThreadStateException	1731
6.333.1.4 IllegalThreadStateException	1731
6.333.1.5 IllegalThreadStateException	1731

6.333.1.6	IllegalThreadStateException	1731
6.333.1.7	~IllegalThreadStateException	1732
6.333.2	Member Function Documentation	1732
6.333.2.1	clone	1732
6.334	activemq::transport::inactivity::InactivityMonitor Class Reference	1733
6.334.1	Constructor & Destructor Documentation	1734
6.334.1.1	InactivityMonitor	1734
6.334.1.2	InactivityMonitor	1734
6.334.1.3	~InactivityMonitor	1734
6.334.2	Member Function Documentation	1734
6.334.2.1	close	1734
6.334.2.2	getInitialDelayTime	1734
6.334.2.3	getReadCheckTime	1734
6.334.2.4	getWriteCheckTime	1734
6.334.2.5	isKeepAliveResponseRequired	1734
6.334.2.6	onCommand	1734
6.334.2.7	oneway	1735
6.334.2.8	onException	1735
6.334.2.9	setInitialDelayTime	1736
6.334.2.10	setKeepAliveResponseRequired	1736
6.334.2.11	setReadCheckTime	1736
6.334.2.12	setWriteCheckTime	1736
6.334.3	Friends And Related Function Documentation	1736
6.334.3.1	AsyncSignalReadErrorTask	1736
6.334.3.2	AsyncWriteTask	1736
6.334.3.3	ReadChecker	1736
6.334.3.4	WriteChecker	1736
6.335	decaf::lang::exceptions::IndexOutOfBoundsException Class Reference	1737
6.335.1	Constructor & Destructor Documentation	1737
6.335.1.1	IndexOutOfBoundsException	1737
6.335.1.2	IndexOutOfBoundsException	1737
6.335.1.3	IndexOutOfBoundsException	1738
6.335.1.4	IndexOutOfBoundsException	1738
6.335.1.5	IndexOutOfBoundsException	1738
6.335.1.6	IndexOutOfBoundsException	1738
6.335.1.7	~IndexOutOfBoundsException	1739

6.335.2 Member Function Documentation	1739
6.335.2.1 clone	1739
6.336decaf::io::InputStream Class Reference	1740
6.336.1 Detailed Description	1740
6.336.2 Constructor & Destructor Documentation	1741
6.336.2.1 ~InputStream	1741
6.336.3 Member Function Documentation	1741
6.336.3.1 available	1741
6.336.3.2 mark	1741
6.336.3.3 markSupported	1741
6.336.3.4 read	1742
6.336.3.5 read	1742
6.336.3.6 reset	1742
6.336.3.7 skip	1743
6.337decaf::internal::nio::IntArrayBuffer Class Reference	1744
6.337.1 Constructor & Destructor Documentation	1745
6.337.1.1 IntArrayBuffer	1745
6.337.1.2 IntArrayBuffer	1745
6.337.1.3 IntArrayBuffer	1746
6.337.1.4 IntArrayBuffer	1746
6.337.1.5 ~IntArrayBuffer	1746
6.337.2 Member Function Documentation	1746
6.337.2.1 array	1746
6.337.2.2 arrayOffset	1747
6.337.2.3 asReadOnlyBuffer	1747
6.337.2.4 compact	1747
6.337.2.5 duplicate	1748
6.337.2.6 get	1748
6.337.2.7 get	1748
6.337.2.8 hasArray	1749
6.337.2.9 isReadOnly	1749
6.337.2.10put	1749
6.337.2.11put	1750
6.337.2.12setReadOnly	1750
6.337.2.13slice	1750
6.338decaf::nio::IntBuffer Class Reference	1751

6.338.1 Detailed Description	1753
6.338.2 Constructor & Destructor Documentation	1753
6.338.2.1 IntBuffer	1753
6.338.2.2 ~IntBuffer	1753
6.338.3 Member Function Documentation	1753
6.338.3.1 allocate	1753
6.338.3.2 array	1753
6.338.3.3 arrayOffset	1754
6.338.3.4 asReadOnlyBuffer	1754
6.338.3.5 compact	1754
6.338.3.6 compareTo	1755
6.338.3.7 duplicate	1755
6.338.3.8 equals	1755
6.338.3.9 get	1756
6.338.3.10get	1756
6.338.3.11get	1756
6.338.3.12get	1757
6.338.3.13hasArray	1757
6.338.3.14operator<	1757
6.338.3.15operator==	1758
6.338.3.16put	1758
6.338.3.17put	1758
6.338.3.18put	1759
6.338.3.19put	1759
6.338.3.20put	1759
6.338.3.21slice	1760
6.338.3.22toString	1760
6.338.3.23wrap	1760
6.338.3.24wrap	1761
6.339decaf::lang::Integer Class Reference	1762
6.339.1 Constructor & Destructor Documentation	1764
6.339.1.1 Integer	1764
6.339.1.2 Integer	1765
6.339.1.3 ~Integer	1765
6.339.2 Member Function Documentation	1765
6.339.2.1 bitCount	1765

6.339.2.2	byteValue	1765
6.339.2.3	compareTo	1765
6.339.2.4	compareTo	1766
6.339.2.5	decode	1766
6.339.2.6	doubleValue	1766
6.339.2.7	equals	1767
6.339.2.8	equals	1767
6.339.2.9	float Value	1767
6.339.2.10	highestOneBit	1767
6.339.2.11	int Value	1767
6.339.2.12	long Value	1768
6.339.2.13	lowestOneBit	1768
6.339.2.14	numberOfLeadingZeros	1768
6.339.2.15	numberOfTrailingZeros	1769
6.339.2.16	operator<	1769
6.339.2.17	operator<	1769
6.339.2.18	operator==	1769
6.339.2.19	operator==	1770
6.339.2.20	parseInt	1770
6.339.2.21	parseInt	1770
6.339.2.22	reverse	1771
6.339.2.23	reverseBytes	1771
6.339.2.24	rotateLeft	1771
6.339.2.25	rotateRight	1772
6.339.2.26	short Value	1772
6.339.2.27	signum	1772
6.339.2.28	toBinaryString	1773
6.339.2.29	toHexString	1773
6.339.2.30	toOctalString	1773
6.339.2.31	toString	1774
6.339.2.32	toString	1774
6.339.2.33	toString	1774
6.339.2.34	valueOf	1774
6.339.2.35	valueOf	1775
6.339.2.36	valueOf	1775
6.339.3	Field Documentation	1775

6.339.3.1 MAX_VALUE	1775
6.339.3.2 MIN_VALUE	1776
6.339.3.3 SIZE	1776
6.340activemq::commands::IntegerResponse Class Reference	1777
6.340.1 Constructor & Destructor Documentation	1778
6.340.1.1 IntegerResponse	1778
6.340.1.2 IntegerResponse	1778
6.340.1.3 ~IntegerResponse	1778
6.340.2 Member Function Documentation	1778
6.340.2.1 cloneDataStructure	1778
6.340.2.2 copyDataStructure	1778
6.340.2.3 equals	1778
6.340.2.4 getDataStructureType	1778
6.340.2.5 getResult	1779
6.340.2.6 operator=	1779
6.340.2.7 setResult	1779
6.340.2.8 toString	1779
6.340.3 Field Documentation	1779
6.340.3.1 ID_INTEGERRESPONSE	1779
6.340.3.2 result	1779
6.341activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller Class Reference	1780
6.341.1 Detailed Description	1780
6.341.2 Constructor & Destructor Documentation	1781
6.341.2.1 IntegerResponseMarshaller	1781
6.341.2.2 ~IntegerResponseMarshaller	1781
6.341.3 Member Function Documentation	1781
6.341.3.1 createObject	1781
6.341.3.2 getDataStructureType	1781
6.341.3.3 looseMarshal	1781
6.341.3.4 looseUnmarshal	1782
6.341.3.5 tightMarshal1	1782
6.341.3.6 tightMarshal2	1783
6.341.3.7 tightUnmarshal	1783
6.342activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference	1784
6.342.1 Detailed Description	1784

6.342.2 Constructor & Destructor Documentation	1785
6.342.2.1 IntegerResponseMarshaller	1785
6.342.2.2 ~IntegerResponseMarshaller	1785
6.342.3 Member Function Documentation	1785
6.342.3.1 createObject	1785
6.342.3.2 getDataStructureType	1785
6.342.3.3 looseMarshal	1785
6.342.3.4 looseUnmarshal	1786
6.342.3.5 tightMarshal1	1786
6.342.3.6 tightMarshal2	1787
6.342.3.7 tightUnmarshal	1787
6.343activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class	
Reference	1788
6.343.1 Detailed Description	1788
6.343.2 Constructor & Destructor Documentation	1789
6.343.2.1 IntegerResponseMarshaller	1789
6.343.2.2 ~IntegerResponseMarshaller	1789
6.343.3 Member Function Documentation	1789
6.343.3.1 createObject	1789
6.343.3.2 getDataStructureType	1789
6.343.3.3 looseMarshal	1789
6.343.3.4 looseUnmarshal	1790
6.343.3.5 tightMarshal1	1790
6.343.3.6 tightMarshal2	1791
6.343.3.7 tightUnmarshal	1791
6.344activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller Class	
Reference	1792
6.344.1 Detailed Description	1792
6.344.2 Constructor & Destructor Documentation	1793
6.344.2.1 IntegerResponseMarshaller	1793
6.344.2.2 ~IntegerResponseMarshaller	1793
6.344.3 Member Function Documentation	1793
6.344.3.1 createObject	1793
6.344.3.2 getDataStructureType	1793
6.344.3.3 looseMarshal	1793
6.344.3.4 looseUnmarshal	1794
6.344.3.5 tightMarshal1	1794

6.344.3.6	tightMarshal2	1795
6.344.3.7	tightUnmarshal	1795
6.345	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class Reference	1796
6.345.1	Detailed Description	1796
6.345.2	Constructor & Destructor Documentation	1797
6.345.2.1	IntegerResponseMarshaller	1797
6.345.2.2	~IntegerResponseMarshaller	1797
6.345.3	Member Function Documentation	1797
6.345.3.1	createObject	1797
6.345.3.2	getDataStructureType	1797
6.345.3.3	looseMarshal	1797
6.345.3.4	looseUnmarshal	1798
6.345.3.5	tightMarshal1	1798
6.345.3.6	tightMarshal2	1799
6.345.3.7	tightUnmarshal	1799
6.346	activemq::transport::mock::InternalCommandListener Class Reference	1800
6.346.1	Detailed Description	1800
6.346.2	Constructor & Destructor Documentation	1800
6.346.2.1	InternalCommandListener	1800
6.346.2.2	~InternalCommandListener	1800
6.346.3	Member Function Documentation	1800
6.346.3.1	onCommand	1800
6.346.3.2	run	1801
6.346.3.3	setResponseBuilder	1801
6.346.3.4	setTransport	1801
6.347	decaf::lang::exceptions::InterruptedException Class Reference	1802
6.347.1	Constructor & Destructor Documentation	1802
6.347.1.1	InterruptedException	1802
6.347.1.2	InterruptedException	1802
6.347.1.3	InterruptedException	1803
6.347.1.4	InterruptedException	1803
6.347.1.5	InterruptedException	1803
6.347.1.6	InterruptedException	1803
6.347.1.7	~InterruptedException	1804
6.347.2	Member Function Documentation	1804
6.347.2.1	clone	1804

6.348	decaf::io::InterruptedIOException Class Reference	1805
6.348.1	Constructor & Destructor Documentation	1805
6.348.1.1	InterruptedIOException	1805
6.348.1.2	InterruptedIOException	1805
6.348.1.3	InterruptedIOException	1806
6.348.1.4	InterruptedIOException	1806
6.348.1.5	InterruptedIOException	1806
6.348.1.6	InterruptedIOException	1806
6.348.1.7	~InterruptedIOException	1807
6.348.2	Member Function Documentation	1807
6.348.2.1	clone	1807
6.349	cms::InvalidClientIdException Class Reference	1808
6.349.1	Detailed Description	1808
6.349.2	Constructor & Destructor Documentation	1808
6.349.2.1	InvalidClientIdException	1808
6.349.2.2	InvalidClientIdException	1808
6.349.2.3	InvalidClientIdException	1808
6.349.2.4	InvalidClientIdException	1808
6.349.2.5	~InvalidClientIdException	1808
6.350	cms::InvalidDestinationException Class Reference	1809
6.350.1	Detailed Description	1809
6.350.2	Constructor & Destructor Documentation	1809
6.350.2.1	InvalidDestinationException	1809
6.350.2.2	InvalidDestinationException	1809
6.350.2.3	InvalidDestinationException	1809
6.350.2.4	InvalidDestinationException	1809
6.350.2.5	~InvalidDestinationException	1809
6.351	decaf::security::InvalidKeyException Class Reference	1810
6.351.1	Constructor & Destructor Documentation	1810
6.351.1.1	InvalidKeyException	1810
6.351.1.2	InvalidKeyException	1810
6.351.1.3	InvalidKeyException	1811
6.351.1.4	InvalidKeyException	1811
6.351.1.5	InvalidKeyException	1811
6.351.1.6	InvalidKeyException	1811
6.351.1.7	~InvalidKeyException	1812

6.351.2 Member Function Documentation	1812
6.351.2.1 clone	1812
6.352decaf::nio::InvalidMarkException Class Reference	1813
6.352.1 Constructor & Destructor Documentation	1813
6.352.1.1 InvalidMarkException	1813
6.352.1.2 InvalidMarkException	1813
6.352.1.3 InvalidMarkException	1814
6.352.1.4 InvalidMarkException	1814
6.352.1.5 InvalidMarkException	1814
6.352.1.6 InvalidMarkException	1814
6.352.1.7 ~InvalidMarkException	1815
6.352.2 Member Function Documentation	1815
6.352.2.1 clone	1815
6.353cms::InvalidSelectorException Class Reference	1816
6.353.1 Detailed Description	1816
6.353.2 Constructor & Destructor Documentation	1816
6.353.2.1 InvalidSelectorException	1816
6.353.2.2 InvalidSelectorException	1816
6.353.2.3 InvalidSelectorException	1816
6.353.2.4 InvalidSelectorException	1816
6.353.2.5 ~InvalidSelectorException	1816
6.354decaf::lang::exceptions::InvalidStateException Class Reference	1817
6.354.1 Constructor & Destructor Documentation	1817
6.354.1.1 InvalidStateException	1817
6.354.1.2 InvalidStateException	1817
6.354.1.3 InvalidStateException	1818
6.354.1.4 InvalidStateException	1818
6.354.1.5 InvalidStateException	1818
6.354.1.6 InvalidStateException	1818
6.354.1.7 ~InvalidStateException	1819
6.354.2 Member Function Documentation	1819
6.354.2.1 clone	1819
6.355decaf::io::IOException Class Reference	1820
6.355.1 Constructor & Destructor Documentation	1820
6.355.1.1 IOException	1820
6.355.1.2 IOException	1820

6.355.1.3	IOException	1821
6.355.1.4	IOException	1821
6.355.1.5	IOException	1821
6.355.1.6	IOException	1821
6.355.1.7	~IOException	1822
6.355.2	Member Function Documentation	1822
6.355.2.1	clone	1822
6.356	activemq::transport::IOTransport Class Reference	1823
6.356.1	Detailed Description	1824
6.356.2	Constructor & Destructor Documentation	1824
6.356.2.1	IOTransport	1824
6.356.2.2	IOTransport	1824
6.356.2.3	~IOTransport	1825
6.356.3	Member Function Documentation	1825
6.356.3.1	close	1825
6.356.3.2	getRemoteAddress	1825
6.356.3.3	getTransportListener	1825
6.356.3.4	isClosed	1825
6.356.3.5	isConnected	1826
6.356.3.6	isFaultTolerant	1826
6.356.3.7	narrow	1826
6.356.3.8	oneway	1826
6.356.3.9	reconnect	1827
6.356.3.10	request	1827
6.356.3.11	request	1827
6.356.3.12	run	1828
6.356.3.13	setInputStream	1828
6.356.3.14	setOutputStream	1828
6.356.3.15	setTransportListener	1828
6.356.3.16	setWireFormat	1828
6.356.3.17	start	1828
6.356.3.18	stop	1829
6.357	decaf::lang::Iterable< E > Class Template Reference	1830
6.357.1	Detailed Description	1830
6.357.2	Constructor & Destructor Documentation	1830
6.357.2.1	~Iterable	1830

6.357.3 Member Function Documentation	1830
6.357.3.1 iterator	1830
6.357.3.2 iterator	1830
6.358 <code>decaf::util::Iterator< T ></code> Class Template Reference	1832
6.358.1 Detailed Description	1832
6.358.2 Constructor & Destructor Documentation	1832
6.358.2.1 <code>~Iterator</code>	1832
6.358.3 Member Function Documentation	1832
6.358.3.1 <code>hasNext</code>	1832
6.358.3.2 <code>next</code>	1832
6.358.3.3 <code>remove</code>	1833
6.359 <code>activemq::commands::JournalQueueAck</code> Class Reference	1834
6.359.1 Constructor & Destructor Documentation	1835
6.359.1.1 <code>JournalQueueAck</code>	1835
6.359.1.2 <code>JournalQueueAck</code>	1835
6.359.1.3 <code>~JournalQueueAck</code>	1835
6.359.2 Member Function Documentation	1835
6.359.2.1 <code>cloneDataStructure</code>	1835
6.359.2.2 <code>copyDataStructure</code>	1835
6.359.2.3 <code>equals</code>	1835
6.359.2.4 <code>getDataStructureType</code>	1835
6.359.2.5 <code>getDestination</code>	1836
6.359.2.6 <code>getDestination</code>	1836
6.359.2.7 <code>getMessageAck</code>	1836
6.359.2.8 <code>getMessageAck</code>	1836
6.359.2.9 <code>operator=</code>	1836
6.359.2.10 <code>setDestination</code>	1836
6.359.2.11 <code>setMessageAck</code>	1836
6.359.2.12 <code>toString</code>	1836
6.359.3 Field Documentation	1837
6.359.3.1 <code>destination</code>	1837
6.359.3.2 <code>ID_JOURNALQUEUEACK</code>	1837
6.359.3.3 <code>messageAck</code>	1837
6.360 <code>activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller</code> Class Reference	1838
6.360.1 Detailed Description	1838
6.360.2 Constructor & Destructor Documentation	1839

6.360.2.1 JournalQueueAckMarshaller	1839
6.360.2.2 ~JournalQueueAckMarshaller	1839
6.360.3 Member Function Documentation	1839
6.360.3.1 createObject	1839
6.360.3.2 getDataStructureType	1839
6.360.3.3 looseMarshal	1839
6.360.3.4 looseUnmarshal	1840
6.360.3.5 tightMarshal1	1840
6.360.3.6 tightMarshal2	1840
6.360.3.7 tightUnmarshal	1841
6.361activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller Class	
Reference	1842
6.361.1 Detailed Description	1842
6.361.2 Constructor & Destructor Documentation	1843
6.361.2.1 JournalQueueAckMarshaller	1843
6.361.2.2 ~JournalQueueAckMarshaller	1843
6.361.3 Member Function Documentation	1843
6.361.3.1 createObject	1843
6.361.3.2 getDataStructureType	1843
6.361.3.3 looseMarshal	1843
6.361.3.4 looseUnmarshal	1844
6.361.3.5 tightMarshal1	1844
6.361.3.6 tightMarshal2	1844
6.361.3.7 tightUnmarshal	1845
6.362activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller Class	
Reference	1846
6.362.1 Detailed Description	1846
6.362.2 Constructor & Destructor Documentation	1847
6.362.2.1 JournalQueueAckMarshaller	1847
6.362.2.2 ~JournalQueueAckMarshaller	1847
6.362.3 Member Function Documentation	1847
6.362.3.1 createObject	1847
6.362.3.2 getDataStructureType	1847
6.362.3.3 looseMarshal	1847
6.362.3.4 looseUnmarshal	1848
6.362.3.5 tightMarshal1	1848
6.362.3.6 tightMarshal2	1848

6.362.3.7 tightUnmarshal	1849
6.363activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class	
Reference	1850
6.363.1 Detailed Description	1850
6.363.2 Constructor & Destructor Documentation	1851
6.363.2.1 JournalQueueAckMarshaller	1851
6.363.2.2 ~JournalQueueAckMarshaller	1851
6.363.3 Member Function Documentation	1851
6.363.3.1 createObject	1851
6.363.3.2 getDataStructureType	1851
6.363.3.3 looseMarshal	1851
6.363.3.4 looseUnmarshal	1852
6.363.3.5 tightMarshal1	1852
6.363.3.6 tightMarshal2	1852
6.363.3.7 tightUnmarshal	1853
6.364activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller Class	
Reference	1854
6.364.1 Detailed Description	1854
6.364.2 Constructor & Destructor Documentation	1855
6.364.2.1 JournalQueueAckMarshaller	1855
6.364.2.2 ~JournalQueueAckMarshaller	1855
6.364.3 Member Function Documentation	1855
6.364.3.1 createObject	1855
6.364.3.2 getDataStructureType	1855
6.364.3.3 looseMarshal	1855
6.364.3.4 looseUnmarshal	1856
6.364.3.5 tightMarshal1	1856
6.364.3.6 tightMarshal2	1856
6.364.3.7 tightUnmarshal	1857
6.365activemq::commands::JournalTopicAck Class Reference	1858
6.365.1 Constructor & Destructor Documentation	1859
6.365.1.1 JournalTopicAck	1859
6.365.1.2 JournalTopicAck	1859
6.365.1.3 ~JournalTopicAck	1859
6.365.2 Member Function Documentation	1859
6.365.2.1 cloneDataStructure	1859
6.365.2.2 copyDataStructure	1859

6.365.2.3 equals	1860
6.365.2.4 getClientId	1860
6.365.2.5 getClientId	1860
6.365.2.6 getDataStructureType	1860
6.365.2.7 getDestination	1861
6.365.2.8 getDestination	1861
6.365.2.9 getMessageId	1861
6.365.2.10getMessageId	1861
6.365.2.11getMessageSequenceId	1861
6.365.2.12getSubscriptionName	1861
6.365.2.13getSubscriptionName	1861
6.365.2.14getTransactionId	1861
6.365.2.15getTransactionId	1861
6.365.2.16operator=	1861
6.365.2.17setClientId	1861
6.365.2.18setDestination	1861
6.365.2.19setMessageId	1861
6.365.2.20setMessageSequenceId	1861
6.365.2.21setSubscriptionName	1861
6.365.2.22setTransactionId	1861
6.365.2.23toString	1861
6.365.3 Field Documentation	1862
6.365.3.1 clientId	1862
6.365.3.2 destination	1862
6.365.3.3 ID_JOURNALTOPICACK	1862
6.365.3.4 messageId	1862
6.365.3.5 messageSequenceId	1862
6.365.3.6 subscriptionName	1862
6.365.3.7 transactionId	1862
6.366activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	Class
Reference	1863
6.366.1 Detailed Description	1863
6.366.2 Constructor & Destructor Documentation	1864
6.366.2.1 JournalTopicAckMarshaller	1864
6.366.2.2 ~JournalTopicAckMarshaller	1864
6.366.3 Member Function Documentation	1864
6.366.3.1 createObject	1864

6.366.3.2	getDataStructureType	1864
6.366.3.3	looseMarshal	1864
6.366.3.4	looseUnmarshal	1865
6.366.3.5	tightMarshal1	1865
6.366.3.6	tightMarshal2	1865
6.366.3.7	tightUnmarshal	1866
6.367	activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller Class	
	Reference	1867
6.367.1	Detailed Description	1867
6.367.2	Constructor & Destructor Documentation	1868
6.367.2.1	JournalTopicAckMarshaller	1868
6.367.2.2	~JournalTopicAckMarshaller	1868
6.367.3	Member Function Documentation	1868
6.367.3.1	createObject	1868
6.367.3.2	getDataStructureType	1868
6.367.3.3	looseMarshal	1868
6.367.3.4	looseUnmarshal	1869
6.367.3.5	tightMarshal1	1869
6.367.3.6	tightMarshal2	1869
6.367.3.7	tightUnmarshal	1870
6.368	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class	
	Reference	1871
6.368.1	Detailed Description	1871
6.368.2	Constructor & Destructor Documentation	1872
6.368.2.1	JournalTopicAckMarshaller	1872
6.368.2.2	~JournalTopicAckMarshaller	1872
6.368.3	Member Function Documentation	1872
6.368.3.1	createObject	1872
6.368.3.2	getDataStructureType	1872
6.368.3.3	looseMarshal	1872
6.368.3.4	looseUnmarshal	1873
6.368.3.5	tightMarshal1	1873
6.368.3.6	tightMarshal2	1873
6.368.3.7	tightUnmarshal	1874
6.369	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller Class	
	Reference	1875
6.369.1	Detailed Description	1875

6.369.2 Constructor & Destructor Documentation	1876
6.369.2.1 JournalTopicAckMarshaller	1876
6.369.2.2 ~JournalTopicAckMarshaller	1876
6.369.3 Member Function Documentation	1876
6.369.3.1 createObject	1876
6.369.3.2 getDataStructureType	1876
6.369.3.3 looseMarshal	1876
6.369.3.4 looseUnmarshal	1877
6.369.3.5 tightMarshal1	1877
6.369.3.6 tightMarshal2	1877
6.369.3.7 tightUnmarshal	1878
6.370activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class Reference	1879
6.370.1 Detailed Description	1879
6.370.2 Constructor & Destructor Documentation	1880
6.370.2.1 JournalTopicAckMarshaller	1880
6.370.2.2 ~JournalTopicAckMarshaller	1880
6.370.3 Member Function Documentation	1880
6.370.3.1 createObject	1880
6.370.3.2 getDataStructureType	1880
6.370.3.3 looseMarshal	1880
6.370.3.4 looseUnmarshal	1881
6.370.3.5 tightMarshal1	1881
6.370.3.6 tightMarshal2	1881
6.370.3.7 tightUnmarshal	1882
6.371activemq::commands::JournalTrace Class Reference	1883
6.371.1 Constructor & Destructor Documentation	1884
6.371.1.1 JournalTrace	1884
6.371.1.2 JournalTrace	1884
6.371.1.3 ~JournalTrace	1884
6.371.2 Member Function Documentation	1884
6.371.2.1 cloneDataStructure	1884
6.371.2.2 copyDataStructure	1884
6.371.2.3 equals	1884
6.371.2.4 getDataStructureType	1884
6.371.2.5 getMessage	1885
6.371.2.6 getMessage	1885

6.371.2.7 operator=	1885
6.371.2.8 setMessage	1885
6.371.2.9 toString	1885
6.371.3 Field Documentation	1885
6.371.3.1 ID_JOURNALTRACE	1885
6.371.3.2 message	1885
6.372activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference	1886
6.372.1 Detailed Description	1886
6.372.2 Constructor & Destructor Documentation	1887
6.372.2.1 JournalTraceMarshaller	1887
6.372.2.2 ~JournalTraceMarshaller	1887
6.372.3 Member Function Documentation	1887
6.372.3.1 createObject	1887
6.372.3.2 getDataStructureType	1887
6.372.3.3 looseMarshal	1887
6.372.3.4 looseUnmarshal	1888
6.372.3.5 tightMarshal1	1888
6.372.3.6 tightMarshal2	1888
6.372.3.7 tightUnmarshal	1889
6.373activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference	1890
6.373.1 Detailed Description	1890
6.373.2 Constructor & Destructor Documentation	1891
6.373.2.1 JournalTraceMarshaller	1891
6.373.2.2 ~JournalTraceMarshaller	1891
6.373.3 Member Function Documentation	1891
6.373.3.1 createObject	1891
6.373.3.2 getDataStructureType	1891
6.373.3.3 looseMarshal	1891
6.373.3.4 looseUnmarshal	1892
6.373.3.5 tightMarshal1	1892
6.373.3.6 tightMarshal2	1892
6.373.3.7 tightUnmarshal	1893
6.374activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference	1894
6.374.1 Detailed Description	1894

6.374.2 Constructor & Destructor Documentation	1895
6.374.2.1 JournalTraceMarshaller	1895
6.374.2.2 ~JournalTraceMarshaller	1895
6.374.3 Member Function Documentation	1895
6.374.3.1 createObject	1895
6.374.3.2 getDataStructureType	1895
6.374.3.3 looseMarshal	1895
6.374.3.4 looseUnmarshal	1896
6.374.3.5 tightMarshal1	1896
6.374.3.6 tightMarshal2	1896
6.374.3.7 tightUnmarshal	1897
6.375activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference	1898
6.375.1 Detailed Description	1898
6.375.2 Constructor & Destructor Documentation	1899
6.375.2.1 JournalTraceMarshaller	1899
6.375.2.2 ~JournalTraceMarshaller	1899
6.375.3 Member Function Documentation	1899
6.375.3.1 createObject	1899
6.375.3.2 getDataStructureType	1899
6.375.3.3 looseMarshal	1899
6.375.3.4 looseUnmarshal	1900
6.375.3.5 tightMarshal1	1900
6.375.3.6 tightMarshal2	1900
6.375.3.7 tightUnmarshal	1901
6.376activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference	1902
6.376.1 Detailed Description	1902
6.376.2 Constructor & Destructor Documentation	1903
6.376.2.1 JournalTraceMarshaller	1903
6.376.2.2 ~JournalTraceMarshaller	1903
6.376.3 Member Function Documentation	1903
6.376.3.1 createObject	1903
6.376.3.2 getDataStructureType	1903
6.376.3.3 looseMarshal	1903
6.376.3.4 looseUnmarshal	1904
6.376.3.5 tightMarshal1	1904

6.376.3.6 tightMarshal2	1904
6.376.3.7 tightUnmarshal	1905
6.377activemq::commands::JournalTransaction Class Reference	1906
6.377.1 Constructor & Destructor Documentation	1907
6.377.1.1 JournalTransaction	1907
6.377.1.2 JournalTransaction	1907
6.377.1.3 ~JournalTransaction	1907
6.377.2 Member Function Documentation	1907
6.377.2.1 cloneDataStructure	1907
6.377.2.2 copyDataStructure	1907
6.377.2.3 equals	1907
6.377.2.4 getDataStructureType	1908
6.377.2.5 getTransactionId	1908
6.377.2.6 getTransactionId	1908
6.377.2.7 getType	1908
6.377.2.8 getWasPrepared	1908
6.377.2.9 operator=	1908
6.377.2.10setTransactionId	1908
6.377.2.11setType	1908
6.377.2.12setWasPrepared	1908
6.377.2.13oString	1908
6.377.3 Field Documentation	1909
6.377.3.1 ID_ JOURNALTRANSACTION	1909
6.377.3.2 transactionId	1909
6.377.3.3 type	1909
6.377.3.4 wasPrepared	1909
6.378activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class Reference	1910
6.378.1 Detailed Description	1910
6.378.2 Constructor & Destructor Documentation	1911
6.378.2.1 JournalTransactionMarshaller	1911
6.378.2.2 ~JournalTransactionMarshaller	1911
6.378.3 Member Function Documentation	1911
6.378.3.1 createObject	1911
6.378.3.2 getDataStructureType	1911
6.378.3.3 looseMarshal	1911
6.378.3.4 looseUnmarshal	1912

6.378.3.5 tightMarshal1	1912
6.378.3.6 tightMarshal2	1912
6.378.3.7 tightUnmarshal	1913
6.379activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class	
Reference	1914
6.379.1 Detailed Description	1914
6.379.2 Constructor & Destructor Documentation	1915
6.379.2.1 JournalTransactionMarshaller	1915
6.379.2.2 ~JournalTransactionMarshaller	1915
6.379.3 Member Function Documentation	1915
6.379.3.1 createObject	1915
6.379.3.2 getDataStructureType	1915
6.379.3.3 looseMarshal	1915
6.379.3.4 looseUnmarshal	1916
6.379.3.5 tightMarshal1	1916
6.379.3.6 tightMarshal2	1916
6.379.3.7 tightUnmarshal	1917
6.380activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller Class	
Reference	1918
6.380.1 Detailed Description	1918
6.380.2 Constructor & Destructor Documentation	1919
6.380.2.1 JournalTransactionMarshaller	1919
6.380.2.2 ~JournalTransactionMarshaller	1919
6.380.3 Member Function Documentation	1919
6.380.3.1 createObject	1919
6.380.3.2 getDataStructureType	1919
6.380.3.3 looseMarshal	1919
6.380.3.4 looseUnmarshal	1920
6.380.3.5 tightMarshal1	1920
6.380.3.6 tightMarshal2	1920
6.380.3.7 tightUnmarshal	1921
6.381activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class	
Reference	1922
6.381.1 Detailed Description	1922
6.381.2 Constructor & Destructor Documentation	1923
6.381.2.1 JournalTransactionMarshaller	1923
6.381.2.2 ~JournalTransactionMarshaller	1923

6.381.3 Member Function Documentation	1923
6.381.3.1 createObject	1923
6.381.3.2 getDataStructureType	1923
6.381.3.3 looseMarshal	1923
6.381.3.4 looseUnmarshal	1924
6.381.3.5 tightMarshal1	1924
6.381.3.6 tightMarshal2	1924
6.381.3.7 tightUnmarshal	1925
6.382activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class Reference	1926
6.382.1 Detailed Description	1926
6.382.2 Constructor & Destructor Documentation	1927
6.382.2.1 JournalTransactionMarshaller	1927
6.382.2.2 ~JournalTransactionMarshaller	1927
6.382.3 Member Function Documentation	1927
6.382.3.1 createObject	1927
6.382.3.2 getDataStructureType	1927
6.382.3.3 looseMarshal	1927
6.382.3.4 looseUnmarshal	1928
6.382.3.5 tightMarshal1	1928
6.382.3.6 tightMarshal2	1928
6.382.3.7 tightUnmarshal	1929
6.383activemq::commands::KeepAliveInfo Class Reference	1930
6.383.1 Constructor & Destructor Documentation	1931
6.383.1.1 KeepAliveInfo	1931
6.383.1.2 KeepAliveInfo	1931
6.383.1.3 ~KeepAliveInfo	1931
6.383.2 Member Function Documentation	1931
6.383.2.1 cloneDataStructure	1931
6.383.2.2 copyDataStructure	1931
6.383.2.3 equals	1931
6.383.2.4 getDataStructureType	1931
6.383.2.5 isKeepAliveInfo	1932
6.383.2.6 operator=	1932
6.383.2.7 toString	1932
6.383.2.8 visit	1932
6.383.3 Field Documentation	1932

6.383.3.1 ID_KEEPAALIVEINFO	1932
6.384activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference	1933
6.384.1 Detailed Description	1933
6.384.2 Constructor & Destructor Documentation	1934
6.384.2.1 KeepAliveInfoMarshaller	1934
6.384.2.2 ~KeepAliveInfoMarshaller	1934
6.384.3 Member Function Documentation	1934
6.384.3.1 createObject	1934
6.384.3.2 getDataStructureType	1934
6.384.3.3 looseMarshal	1934
6.384.3.4 looseUnmarshal	1935
6.384.3.5 tightMarshal1	1935
6.384.3.6 tightMarshal2	1935
6.384.3.7 tightUnmarshal	1936
6.385activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference	1937
6.385.1 Detailed Description	1937
6.385.2 Constructor & Destructor Documentation	1938
6.385.2.1 KeepAliveInfoMarshaller	1938
6.385.2.2 ~KeepAliveInfoMarshaller	1938
6.385.3 Member Function Documentation	1938
6.385.3.1 createObject	1938
6.385.3.2 getDataStructureType	1938
6.385.3.3 looseMarshal	1938
6.385.3.4 looseUnmarshal	1939
6.385.3.5 tightMarshal1	1939
6.385.3.6 tightMarshal2	1939
6.385.3.7 tightUnmarshal	1940
6.386activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference	1941
6.386.1 Detailed Description	1941
6.386.2 Constructor & Destructor Documentation	1942
6.386.2.1 KeepAliveInfoMarshaller	1942
6.386.2.2 ~KeepAliveInfoMarshaller	1942
6.386.3 Member Function Documentation	1942
6.386.3.1 createObject	1942

6.386.3.2	getDataStructureType	1942
6.386.3.3	looseMarshal	1942
6.386.3.4	looseUnmarshal	1943
6.386.3.5	tightMarshal1	1943
6.386.3.6	tightMarshal2	1943
6.386.3.7	tightUnmarshal	1944
6.387	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference	1945
6.387.1	Detailed Description	1945
6.387.2	Constructor & Destructor Documentation	1946
6.387.2.1	KeepAliveInfoMarshaller	1946
6.387.2.2	~KeepAliveInfoMarshaller	1946
6.387.3	Member Function Documentation	1946
6.387.3.1	createObject	1946
6.387.3.2	getDataStructureType	1946
6.387.3.3	looseMarshal	1946
6.387.3.4	looseUnmarshal	1947
6.387.3.5	tightMarshal1	1947
6.387.3.6	tightMarshal2	1947
6.387.3.7	tightUnmarshal	1948
6.388	activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference	1949
6.388.1	Detailed Description	1949
6.388.2	Constructor & Destructor Documentation	1950
6.388.2.1	KeepAliveInfoMarshaller	1950
6.388.2.2	~KeepAliveInfoMarshaller	1950
6.388.3	Member Function Documentation	1950
6.388.3.1	createObject	1950
6.388.3.2	getDataStructureType	1950
6.388.3.3	looseMarshal	1950
6.388.3.4	looseUnmarshal	1951
6.388.3.5	tightMarshal1	1951
6.388.3.6	tightMarshal2	1951
6.388.3.7	tightUnmarshal	1952
6.389	decaf::security::Key Class Reference	1953
6.389.1	Detailed Description	1953
6.389.2	Constructor & Destructor Documentation	1954

6.389.2.1 ~Key	1954
6.389.3 Member Function Documentation	1954
6.389.3.1 getAlgorithm	1954
6.389.3.2 getEncoded	1954
6.389.3.3 getFormat	1954
6.390decaf::security::KeyException Class Reference	1955
6.390.1 Constructor & Destructor Documentation	1955
6.390.1.1 KeyException	1955
6.390.1.2 KeyException	1955
6.390.1.3 KeyException	1956
6.390.1.4 KeyException	1956
6.390.1.5 KeyException	1956
6.390.1.6 KeyException	1956
6.390.1.7 ~KeyException	1957
6.390.2 Member Function Documentation	1957
6.390.2.1 clone	1957
6.391activemq::commands::LastPartialCommand Class Reference	1958
6.391.1 Constructor & Destructor Documentation	1959
6.391.1.1 LastPartialCommand	1959
6.391.1.2 LastPartialCommand	1959
6.391.1.3 ~LastPartialCommand	1959
6.391.2 Member Function Documentation	1959
6.391.2.1 cloneDataStructure	1959
6.391.2.2 copyDataStructure	1959
6.391.2.3 equals	1959
6.391.2.4 getDataStructureType	1960
6.391.2.5 operator=	1960
6.391.2.6 toString	1960
6.391.3 Field Documentation	1960
6.391.3.1 ID_LASTPARTIALCOMMAND	1960
6.392activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller Class Reference	1961
6.392.1 Detailed Description	1961
6.392.2 Constructor & Destructor Documentation	1962
6.392.2.1 LastPartialCommandMarshaller	1962
6.392.2.2 ~LastPartialCommandMarshaller	1962
6.392.3 Member Function Documentation	1962

6.392.3.1 createObject	1962
6.392.3.2 getDataStructureType	1962
6.392.3.3 looseMarshal	1962
6.392.3.4 looseUnmarshal	1963
6.392.3.5 tightMarshal1	1963
6.392.3.6 tightMarshal2	1964
6.392.3.7 tightUnmarshal	1964
6.393activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	
Class Reference	1965
6.393.1 Detailed Description	1965
6.393.2 Constructor & Destructor Documentation	1966
6.393.2.1 LastPartialCommandMarshaller	1966
6.393.2.2 ~LastPartialCommandMarshaller	1966
6.393.3 Member Function Documentation	1966
6.393.3.1 createObject	1966
6.393.3.2 getDataStructureType	1966
6.393.3.3 looseMarshal	1966
6.393.3.4 looseUnmarshal	1967
6.393.3.5 tightMarshal1	1967
6.393.3.6 tightMarshal2	1968
6.393.3.7 tightUnmarshal	1968
6.394activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
Class Reference	1969
6.394.1 Detailed Description	1969
6.394.2 Constructor & Destructor Documentation	1970
6.394.2.1 LastPartialCommandMarshaller	1970
6.394.2.2 ~LastPartialCommandMarshaller	1970
6.394.3 Member Function Documentation	1970
6.394.3.1 createObject	1970
6.394.3.2 getDataStructureType	1970
6.394.3.3 looseMarshal	1970
6.394.3.4 looseUnmarshal	1971
6.394.3.5 tightMarshal1	1971
6.394.3.6 tightMarshal2	1972
6.394.3.7 tightUnmarshal	1972
6.395activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
Class Reference	1973

6.395.1 Detailed Description	1973
6.395.2 Constructor & Destructor Documentation	1974
6.395.2.1 LastPartialCommandMarshaller	1974
6.395.2.2 ~LastPartialCommandMarshaller	1974
6.395.3 Member Function Documentation	1974
6.395.3.1 createObject	1974
6.395.3.2 getDataStructureType	1974
6.395.3.3 looseMarshal	1974
6.395.3.4 looseUnmarshal	1975
6.395.3.5 tightMarshal1	1975
6.395.3.6 tightMarshal2	1976
6.395.3.7 tightUnmarshal	1976
6.396activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
Class Reference	1977
6.396.1 Detailed Description	1977
6.396.2 Constructor & Destructor Documentation	1978
6.396.2.1 LastPartialCommandMarshaller	1978
6.396.2.2 ~LastPartialCommandMarshaller	1978
6.396.3 Member Function Documentation	1978
6.396.3.1 createObject	1978
6.396.3.2 getDataStructureType	1978
6.396.3.3 looseMarshal	1978
6.396.3.4 looseUnmarshal	1979
6.396.3.5 tightMarshal1	1979
6.396.3.6 tightMarshal2	1980
6.396.3.7 tightUnmarshal	1980
6.397decaf::util::comparators::Less< E > Class Template Reference	1981
6.397.1 Detailed Description	1981
6.397.2 Constructor & Destructor Documentation	1981
6.397.2.1 Less	1981
6.397.2.2 ~Less	1981
6.397.3 Member Function Documentation	1981
6.397.3.1 compare	1981
6.397.3.2 operator()	1982
6.398std::less< decaf::lang::Pointer< T > > Struct Template Reference	1983
6.398.1 Detailed Description	1983
6.398.2 Member Function Documentation	1983

6.398.2.1 operator()	1983
6.399decaf::util::List< E > Class Template Reference	1984
6.399.1 Detailed Description	1985
6.399.2 Constructor & Destructor Documentation	1985
6.399.2.1 ~List	1985
6.399.3 Member Function Documentation	1985
6.399.3.1 add	1985
6.399.3.2 addAll	1985
6.399.3.3 get	1986
6.399.3.4 indexOf	1986
6.399.3.5 lastIndexOf	1987
6.399.3.6 listIterator	1987
6.399.3.7 listIterator	1988
6.399.3.8 listIterator	1988
6.399.3.9 listIterator	1988
6.399.3.10 remove	1989
6.399.3.11 set	1989
6.400decaf::util::ListIterator< E > Class Template Reference	1990
6.400.1 Detailed Description	1990
6.400.2 Constructor & Destructor Documentation	1991
6.400.2.1 ~ListIterator	1991
6.400.3 Member Function Documentation	1991
6.400.3.1 add	1991
6.400.3.2 hasPrevious	1991
6.400.3.3 nextIndex	1991
6.400.3.4 previous	1992
6.400.3.5 previousIndex	1992
6.400.3.6 set	1992
6.401activemq::commands::LocalTransactionId Class Reference	1993
6.401.1 Member Typedef Documentation	1994
6.401.1.1 COMPARATOR	1994
6.401.2 Constructor & Destructor Documentation	1994
6.401.2.1 LocalTransactionId	1994
6.401.2.2 LocalTransactionId	1994
6.401.2.3 ~LocalTransactionId	1994
6.401.3 Member Function Documentation	1994

6.401.3.1 cloneDataStructure	1994
6.401.3.2 compareTo	1994
6.401.3.3 copyDataStructure	1994
6.401.3.4 equals	1995
6.401.3.5 equals	1995
6.401.3.6 getConnectionId	1995
6.401.3.7 getConnectionId	1995
6.401.3.8 getDataStructureType	1995
6.401.3.9 getValue	1996
6.401.3.10 operator<	1996
6.401.3.11 operator=	1996
6.401.3.12 operator==	1996
6.401.3.13 setConnectionId	1996
6.401.3.14 setValue	1996
6.401.3.15 toString	1996
6.401.4 Field Documentation	1996
6.401.4.1 connectionId	1996
6.401.4.2 ID_LOCALTRANSACTIONID	1996
6.401.4.3 value	1996
6.402activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class	
Reference	1997
6.402.1 Detailed Description	1997
6.402.2 Constructor & Destructor Documentation	1998
6.402.2.1 LocalTransactionIdMarshaller	1998
6.402.2.2 ~LocalTransactionIdMarshaller	1998
6.402.3 Member Function Documentation	1998
6.402.3.1 createObject	1998
6.402.3.2 getDataStructureType	1998
6.402.3.3 looseMarshal	1998
6.402.3.4 looseUnmarshal	1999
6.402.3.5 tightMarshal1	1999
6.402.3.6 tightMarshal2	1999
6.402.3.7 tightUnmarshal	2000
6.403activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class	
Reference	2001
6.403.1 Detailed Description	2001
6.403.2 Constructor & Destructor Documentation	2002

6.403.2.1 LocalTransactionIdMarshaller	2002
6.403.2.2 ~LocalTransactionIdMarshaller	2002
6.403.3 Member Function Documentation	2002
6.403.3.1 createObject	2002
6.403.3.2 getDataStructureType	2002
6.403.3.3 looseMarshal	2002
6.403.3.4 looseUnmarshal	2003
6.403.3.5 tightMarshal1	2003
6.403.3.6 tightMarshal2	2003
6.403.3.7 tightUnmarshal	2004
6.404activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class	
Reference	2005
6.404.1 Detailed Description	2005
6.404.2 Constructor & Destructor Documentation	2006
6.404.2.1 LocalTransactionIdMarshaller	2006
6.404.2.2 ~LocalTransactionIdMarshaller	2006
6.404.3 Member Function Documentation	2006
6.404.3.1 createObject	2006
6.404.3.2 getDataStructureType	2006
6.404.3.3 looseMarshal	2006
6.404.3.4 looseUnmarshal	2007
6.404.3.5 tightMarshal1	2007
6.404.3.6 tightMarshal2	2007
6.404.3.7 tightUnmarshal	2008
6.405activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller Class	
Reference	2009
6.405.1 Detailed Description	2009
6.405.2 Constructor & Destructor Documentation	2010
6.405.2.1 LocalTransactionIdMarshaller	2010
6.405.2.2 ~LocalTransactionIdMarshaller	2010
6.405.3 Member Function Documentation	2010
6.405.3.1 createObject	2010
6.405.3.2 getDataStructureType	2010
6.405.3.3 looseMarshal	2010
6.405.3.4 looseUnmarshal	2011
6.405.3.5 tightMarshal1	2011
6.405.3.6 tightMarshal2	2011

6.405.3.7 tightUnmarshal	2012
6.406activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class Reference	2013
6.406.1 Detailed Description	2013
6.406.2 Constructor & Destructor Documentation	2014
6.406.2.1 LocalTransactionIdMarshaller	2014
6.406.2.2 ~LocalTransactionIdMarshaller	2014
6.406.3 Member Function Documentation	2014
6.406.3.1 createObject	2014
6.406.3.2 getDataStructureType	2014
6.406.3.3 looseMarshal	2014
6.406.3.4 looseUnmarshal	2015
6.406.3.5 tightMarshal1	2015
6.406.3.6 tightMarshal2	2015
6.406.3.7 tightUnmarshal	2016
6.407decaf::util::concurrent::locks::Lock Class Reference	2017
6.407.1 Detailed Description	2017
6.407.2 Constructor & Destructor Documentation	2018
6.407.2.1 ~Lock	2018
6.407.3 Member Function Documentation	2018
6.407.3.1 lock	2018
6.407.3.2 lockInterruptibly	2019
6.407.3.3 newCondition	2019
6.407.3.4 tryLock	2020
6.407.3.5 tryLock	2021
6.407.3.6 unlock	2021
6.408decaf::util::concurrent::Lock Class Reference	2023
6.408.1 Detailed Description	2023
6.408.2 Constructor & Destructor Documentation	2023
6.408.2.1 Lock	2023
6.408.2.2 ~Lock	2024
6.408.3 Member Function Documentation	2024
6.408.3.1 isLocked	2024
6.408.3.2 lock	2024
6.408.3.3 unlock	2024
6.409decaf::util::concurrent::locks::LockSupport Class Reference	2025
6.409.1 Detailed Description	2025

6.409.2 Constructor & Destructor Documentation	2026
6.409.2.1 ~LockSupport	2026
6.409.3 Member Function Documentation	2026
6.409.3.1 park	2026
6.409.3.2 parkNanos	2026
6.409.3.3 parkUntil	2027
6.409.3.4 unpark	2027
6.410decaf::util::logging::Logger Class Reference	2028
6.410.1 Constructor & Destructor Documentation	2029
6.410.1.1 Logger	2029
6.410.1.2 ~Logger	2030
6.410.2 Member Function Documentation	2030
6.410.2.1 addHandler	2030
6.410.2.2 debug	2030
6.410.2.3 entry	2030
6.410.2.4 error	2031
6.410.2.5 exit	2031
6.410.2.6 fatal	2031
6.410.2.7 getAnonymousLogger	2031
6.410.2.8 getFilter	2032
6.410.2.9 getLevel	2032
6.410.2.10getLogger	2032
6.410.2.11getName	2032
6.410.2.12getUseParentHandlers	2033
6.410.2.13info	2033
6.410.2.14sLoggable	2033
6.410.2.15log	2033
6.410.2.16og	2034
6.410.2.17og	2034
6.410.2.18og	2034
6.410.2.19removeHandler	2035
6.410.2.20setFilter	2035
6.410.2.21setLevel	2035
6.410.2.22setUseParentHandlers	2035
6.410.2.23warn	2036
6.411decaf::util::logging::LoggerHierarchy Class Reference	2037

6.411.1 Constructor & Destructor Documentation	2037
6.411.1.1 LoggerHierarchy	2037
6.411.1.2 ~LoggerHierarchy	2037
6.412activemq::io::LoggingInputStream Class Reference	2038
6.412.1 Constructor & Destructor Documentation	2038
6.412.1.1 LoggingInputStream	2038
6.412.1.2 ~LoggingInputStream	2038
6.412.2 Member Function Documentation	2038
6.412.2.1 read	2038
6.412.2.2 read	2039
6.413activemq::io::LoggingOutputStream Class Reference	2040
6.413.1 Detailed Description	2040
6.413.2 Constructor & Destructor Documentation	2040
6.413.2.1 LoggingOutputStream	2040
6.413.2.2 ~LoggingOutputStream	2040
6.413.3 Member Function Documentation	2040
6.413.3.1 write	2040
6.413.3.2 write	2041
6.414activemq::transport::logging::LoggingTransport Class Reference	2042
6.414.1 Detailed Description	2042
6.414.2 Constructor & Destructor Documentation	2042
6.414.2.1 LoggingTransport	2042
6.414.2.2 ~LoggingTransport	2043
6.414.3 Member Function Documentation	2043
6.414.3.1 onCommand	2043
6.414.3.2 oneway	2043
6.414.3.3 request	2043
6.414.3.4 request	2044
6.415decaf::util::logging::LogManager Class Reference	2045
6.415.1 Detailed Description	2046
6.415.2 Constructor & Destructor Documentation	2047
6.415.2.1 ~LogManager	2047
6.415.2.2 LogManager	2047
6.415.2.3 LogManager	2047
6.415.3 Member Function Documentation	2047
6.415.3.1 addPropertyChangeListener	2047

6.415.3.2	destroy	2048
6.415.3.3	getInstance	2048
6.415.3.4	getLogger	2048
6.415.3.5	getLoggerNames	2048
6.415.3.6	getProperties	2048
6.415.3.7	getProperty	2048
6.415.3.8	operator=	2049
6.415.3.9	removePropertyChangeListener	2049
6.415.3.10	returnInstance	2049
6.415.3.11	setProperties	2049
6.416	decaf::util::logging::LogRecord Class Reference	2050
6.416.1	Constructor & Destructor Documentation	2051
6.416.1.1	LogRecord	2051
6.416.1.2	~LogRecord	2051
6.416.2	Member Function Documentation	2051
6.416.2.1	getLevel	2051
6.416.2.2	getLoggerName	2051
6.416.2.3	getMessage	2051
6.416.2.4	getSourceFile	2051
6.416.2.5	getSourceFunction	2052
6.416.2.6	getSourceLine	2052
6.416.2.7	getTimestamp	2052
6.416.2.8	getTreadId	2052
6.416.2.9	setLevel	2052
6.416.2.10	setLoggerName	2052
6.416.2.11	setMessage	2053
6.416.2.12	setSourceFile	2053
6.416.2.13	setSourceFunction	2053
6.416.2.14	setSourceLine	2053
6.416.2.15	setTimestamp	2053
6.416.2.16	setTreadId	2053
6.417	decaf::util::logging::LogWriter Class Reference	2055
6.417.1	Constructor & Destructor Documentation	2055
6.417.1.1	LogWriter	2055
6.417.1.2	~LogWriter	2055
6.417.2	Member Function Documentation	2055

6.417.2.1	destroy	2055
6.417.2.2	getInstance	2055
6.417.2.3	log	2056
6.417.2.4	log	2056
6.417.2.5	returnInstance	2056
6.418	decaf::lang::Long Class Reference	2057
6.418.1	Constructor & Destructor Documentation	2059
6.418.1.1	Long	2059
6.418.1.2	Long	2060
6.418.1.3	~Long	2060
6.418.2	Member Function Documentation	2060
6.418.2.1	bitCount	2060
6.418.2.2	byteValue	2060
6.418.2.3	compareTo	2060
6.418.2.4	compareTo	2061
6.418.2.5	decode	2061
6.418.2.6	doubleValue	2061
6.418.2.7	equals	2061
6.418.2.8	equals	2062
6.418.2.9	float Value	2062
6.418.2.10	highestOneBit	2062
6.418.2.11	int Value	2062
6.418.2.12	long Value	2063
6.418.2.13	lowestOneBit	2063
6.418.2.14	numberOfLeadingZeros	2063
6.418.2.15	numberOfTrailingZeros	2063
6.418.2.16	operator<	2064
6.418.2.17	operator<	2064
6.418.2.18	operator==	2064
6.418.2.19	operator==	2065
6.418.2.20	parseLong	2065
6.418.2.21	parseLong	2065
6.418.2.22	reverse	2066
6.418.2.23	reverseBytes	2066
6.418.2.24	rotateLeft	2066
6.418.2.25	rotateRight	2067

6.418.2.26	shortValue	2067
6.418.2.27	signum	2067
6.418.2.28	oBinaryString	2067
6.418.2.29	oHexString	2068
6.418.2.30	oOctalString	2068
6.418.2.31	toString	2069
6.418.2.32	oString	2069
6.418.2.33	oString	2069
6.418.2.34	valueOf	2069
6.418.2.35	valueOf	2069
6.418.2.36	valueOf	2070
6.418.3	Field Documentation	2070
6.418.3.1	MAX_VALUE	2070
6.418.3.2	MIN_VALUE	2070
6.418.3.3	SIZE	2070
6.419	decaf::nio::LongArrayBuffer Class Reference	2071
6.419.1	Constructor & Destructor Documentation	2072
6.419.1.1	LongArrayBuffer	2072
6.419.1.2	LongArrayBuffer	2072
6.419.1.3	LongArrayBuffer	2073
6.419.1.4	LongArrayBuffer	2073
6.419.1.5	~LongArrayBuffer	2073
6.419.2	Member Function Documentation	2073
6.419.2.1	array	2073
6.419.2.2	arrayOffset	2074
6.419.2.3	asReadOnlyBuffer	2074
6.419.2.4	compact	2074
6.419.2.5	duplicate	2075
6.419.2.6	get	2075
6.419.2.7	get	2076
6.419.2.8	hasArray	2076
6.419.2.9	isReadOnly	2076
6.419.2.10	put	2076
6.419.2.11	put	2077
6.419.2.12	setReadOnly	2077
6.419.2.13	slice	2077

6.420	decaf::nio::LongBuffer Class Reference	2079
6.420.1	Detailed Description	2081
6.420.2	Constructor & Destructor Documentation	2081
6.420.2.1	LongBuffer	2081
6.420.2.2	~LongBuffer	2081
6.420.3	Member Function Documentation	2081
6.420.3.1	allocate	2081
6.420.3.2	array	2082
6.420.3.3	arrayOffset	2082
6.420.3.4	asReadOnlyBuffer	2082
6.420.3.5	compact	2083
6.420.3.6	compareTo	2083
6.420.3.7	duplicate	2083
6.420.3.8	equals	2084
6.420.3.9	get	2084
6.420.3.10	get	2084
6.420.3.11	get	2085
6.420.3.12	get	2085
6.420.3.13	hasArray	2085
6.420.3.14	operator<	2085
6.420.3.15	operator==	2086
6.420.3.16	put	2086
6.420.3.17	put	2086
6.420.3.18	put	2087
6.420.3.19	put	2087
6.420.3.20	put	2088
6.420.3.21	slice	2088
6.420.3.22	toString	2089
6.420.3.23	wrap	2089
6.420.3.24	wrap	2089
6.421	activemq::util::LongSequenceGenerator Class Reference	2090
6.421.1	Detailed Description	2090
6.421.2	Constructor & Destructor Documentation	2090
6.421.2.1	LongSequenceGenerator	2090
6.421.2.2	~LongSequenceGenerator	2090
6.421.3	Member Function Documentation	2090

6.421.3.1	getLastSequenceId	2090
6.421.3.2	getNextSequenceId	2090
6.422	decaf::net::MalformedURLException Class Reference	2091
6.422.1	Constructor & Destructor Documentation	2091
6.422.1.1	MalformedURLException	2091
6.422.1.2	MalformedURLException	2091
6.422.1.3	MalformedURLException	2092
6.422.1.4	MalformedURLException	2092
6.422.1.5	MalformedURLException	2092
6.422.1.6	MalformedURLException	2092
6.422.1.7	~MalformedURLException	2093
6.422.2	Member Function Documentation	2093
6.422.2.1	clone	2093
6.423	decaf::util::Map< K, V, COMPARATOR > Class Template Reference	2094
6.423.1	Detailed Description	2095
6.423.2	Constructor & Destructor Documentation	2095
6.423.2.1	Map	2095
6.423.2.2	~Map	2095
6.423.3	Member Function Documentation	2095
6.423.3.1	clear	2095
6.423.3.2	containsKey	2096
6.423.3.3	containsValue	2097
6.423.3.4	copy	2098
6.423.3.5	equals	2098
6.423.3.6	get	2098
6.423.3.7	get	2099
6.423.3.8	isEmpty	2100
6.423.3.9	keySet	2101
6.423.3.10	put	2102
6.423.3.11	putAll	2102
6.423.3.12	remove	2103
6.423.3.13	size	2104
6.423.3.14	values	2105
6.424	cms::MapMessage Class Reference	2106
6.424.1	Detailed Description	2107
6.424.2	Constructor & Destructor Documentation	2108

6.424.2.1 ~MapMessage	2108
6.424.3 Member Function Documentation	2108
6.424.3.1 getBoolean	2108
6.424.3.2 getByte	2109
6.424.3.3 getBytes	2109
6.424.3.4 getChar	2109
6.424.3.5 getDouble	2110
6.424.3.6 getFloat	2110
6.424.3.7 getInt	2110
6.424.3.8 getLong	2111
6.424.3.9 getMapNames	2111
6.424.3.10 getShort	2111
6.424.3.11 getString	2111
6.424.3.12 itemExists	2112
6.424.3.13 setBoolean	2112
6.424.3.14 setByte	2112
6.424.3.15 setBytes	2113
6.424.3.16 setChar	2113
6.424.3.17 setDouble	2114
6.424.3.18 setFloat	2114
6.424.3.19 setInt	2114
6.424.3.20 setLong	2115
6.424.3.21 setShort	2115
6.424.3.22 setString	2115
6.425 decaf::util::logging::MarkBlockLogger Class Reference	2117
6.425.1 Detailed Description	2117
6.425.2 Constructor & Destructor Documentation	2117
6.425.2.1 MarkBlockLogger	2117
6.425.2.2 ~MarkBlockLogger	2117
6.426 activemq::wireformat::MarshalAware Class Reference	2118
6.426.1 Constructor & Destructor Documentation	2118
6.426.1.1 ~MarshalAware	2118
6.426.2 Member Function Documentation	2118
6.426.2.1 afterMarshal	2118
6.426.2.2 afterUnmarshal	2119
6.426.2.3 beforeMarshal	2119

6.426.2.4 beforeUnmarshal	2119
6.426.2.5 getMarshaledForm	2119
6.426.2.6 isMarshalAware	2120
6.426.2.7 setMarshaledForm	2120
6.427activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference . .	2121
6.427.1 Detailed Description	2121
6.427.2 Constructor & Destructor Documentation	2121
6.427.2.1 ~MarshallerFactory	2121
6.427.3 Member Function Documentation	2121
6.427.3.1 configure	2121
6.428activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference . .	2122
6.428.1 Detailed Description	2122
6.428.2 Constructor & Destructor Documentation	2122
6.428.2.1 ~MarshallerFactory	2122
6.428.3 Member Function Documentation	2122
6.428.3.1 configure	2122
6.429activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference . .	2123
6.429.1 Detailed Description	2123
6.429.2 Constructor & Destructor Documentation	2123
6.429.2.1 ~MarshallerFactory	2123
6.429.3 Member Function Documentation	2123
6.429.3.1 configure	2123
6.430activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference . .	2124
6.430.1 Detailed Description	2124
6.430.2 Constructor & Destructor Documentation	2124
6.430.2.1 ~MarshallerFactory	2124
6.430.3 Member Function Documentation	2124
6.430.3.1 configure	2124
6.431activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference . .	2125
6.431.1 Detailed Description	2125
6.431.2 Constructor & Destructor Documentation	2125
6.431.2.1 ~MarshallerFactory	2125
6.431.3 Member Function Documentation	2125
6.431.3.1 configure	2125
6.432decaf::lang::Math Class Reference	2126
6.432.1 Detailed Description	2128

6.432.2 Constructor & Destructor Documentation	2128
6.432.2.1 Math	2128
6.432.2.2 \sim Math	2128
6.432.3 Member Function Documentation	2128
6.432.3.1 abs	2128
6.432.3.2 abs	2128
6.432.3.3 abs	2128
6.432.3.4 abs	2129
6.432.3.5 ceil	2129
6.432.3.6 floor	2130
6.432.3.7 max	2130
6.432.3.8 max	2131
6.432.3.9 max	2131
6.432.3.10max	2131
6.432.3.11max	2132
6.432.3.12min	2132
6.432.3.13min	2132
6.432.3.14min	2132
6.432.3.15min	2133
6.432.3.16min	2133
6.432.3.17min	2133
6.432.3.18pow	2134
6.432.3.19random	2134
6.432.3.20round	2135
6.432.3.21round	2135
6.432.3.22signum	2136
6.432.3.23signum	2136
6.432.3.24qrt	2137
6.432.3.25toDegrees	2140
6.432.3.26toRadians	2140
6.432.4 Field Documentation	2140
6.432.4.1 E	2140
6.432.4.2 PI	2140
6.433activemq::util::MemoryUsage Class Reference	2141
6.433.1 Constructor & Destructor Documentation	2142
6.433.1.1 MemoryUsage	2142

6.433.1.2	MemoryUsage	2142
6.433.1.3	~MemoryUsage	2142
6.433.2	Member Function Documentation	2142
6.433.2.1	decreaseUsage	2142
6.433.2.2	enqueueUsage	2142
6.433.2.3	getLimit	2142
6.433.2.4	getUsage	2143
6.433.2.5	increaseUsage	2143
6.433.2.6	isFull	2143
6.433.2.7	setLimit	2143
6.433.2.8	setUsage	2143
6.433.2.9	waitForSpace	2143
6.433.2.10	waitForSpace	2144
6.434	activemq::commands::Message Class Reference	2145
6.434.1	Constructor & Destructor Documentation	2149
6.434.1.1	Message	2149
6.434.1.2	Message	2149
6.434.1.3	~Message	2149
6.434.2	Member Function Documentation	2149
6.434.2.1	afterUnmarshal	2149
6.434.2.2	beforeMarshal	2149
6.434.2.3	cloneDataStructure	2149
6.434.2.4	copyDataStructure	2150
6.434.2.5	equals	2150
6.434.2.6	getAckHandler	2151
6.434.2.7	getArrival	2152
6.434.2.8	getBrokerInTime	2152
6.434.2.9	getBrokerOutTime	2152
6.434.2.10	getBrokerPath	2152
6.434.2.11	getBrokerPath	2152
6.434.2.12	getCluster	2152
6.434.2.13	getCluster	2152
6.434.2.14	getContent	2152
6.434.2.15	getContent	2152
6.434.2.16	getCorrelationId	2152
6.434.2.17	getCorrelationId	2152

6.434.2.18	getDataStructure	2152
6.434.2.19	getDataStructure	2152
6.434.2.20	getDataStructureType	2152
6.434.2.21	getDestination	2153
6.434.2.22	getDestination	2153
6.434.2.23	getExpiration	2153
6.434.2.24	getGroupID	2153
6.434.2.25	getGroupID	2153
6.434.2.26	getGroupSequence	2153
6.434.2.27	getMarshaledProperties	2153
6.434.2.28	getMarshaledProperties	2153
6.434.2.29	getMessageId	2153
6.434.2.30	getMessageId	2153
6.434.2.31	getMessageProperties	2153
6.434.2.32	getMessageProperties	2153
6.434.2.33	getOriginalDestination	2154
6.434.2.34	getOriginalDestination	2154
6.434.2.35	getOriginalTransactionId	2154
6.434.2.36	getOriginalTransactionId	2154
6.434.2.37	getPriority	2154
6.434.2.38	getProducerId	2154
6.434.2.39	getProducerId	2154
6.434.2.40	getRedeliveryCounter	2154
6.434.2.41	getReplyTo	2154
6.434.2.42	getReplyTo	2154
6.434.2.43	getSize	2154
6.434.2.44	getTargetConsumerId	2155
6.434.2.45	getTargetConsumerId	2155
6.434.2.46	getTimestamp	2155
6.434.2.47	getTransactionId	2155
6.434.2.48	getTransactionId	2155
6.434.2.49	getType	2155
6.434.2.50	getType	2155
6.434.2.51	getUserID	2155
6.434.2.52	getUserID	2155
6.434.2.53	isCompressed	2155

6.434.2.54sDroppable	2155
6.434.2.55sExpired	2155
6.434.2.56sMarshalAware	2155
6.434.2.57sMessage	2156
6.434.2.58sPersistent	2156
6.434.2.59sReadOnlyBody	2156
6.434.2.60sReadOnlyProperties	2156
6.434.2.61sRecievedByDFBridge	2156
6.434.2.62nSend	2156
6.434.2.63operator=	2157
6.434.2.64setAckHandler	2157
6.434.2.65setArrival	2158
6.434.2.66setBrokerInTime	2158
6.434.2.67setBrokerOutTime	2158
6.434.2.68setBrokerPath	2158
6.434.2.69setCluster	2158
6.434.2.70setCompressed	2158
6.434.2.71setContent	2158
6.434.2.72setCorrelationId	2158
6.434.2.73setDataStructure	2158
6.434.2.74setDestination	2158
6.434.2.75setDroppable	2158
6.434.2.76setExpiration	2158
6.434.2.77setGroupID	2158
6.434.2.78setGroupSequence	2158
6.434.2.79setMarshallledProperties	2158
6.434.2.80setMessageId	2158
6.434.2.81setOriginalDestination	2158
6.434.2.82setOriginalTransactionId	2158
6.434.2.83setPersistent	2158
6.434.2.84setPriority	2158
6.434.2.85setProducerId	2158
6.434.2.86setReadOnlyBody	2158
6.434.2.87setReadOnlyProperties	2159
6.434.2.88setRecievedByDFBridge	2159
6.434.2.89setRedeliveryCounter	2159

6.434.2.90	setReplyTo	2159
6.434.2.91	setTargetConsumerId	2159
6.434.2.92	setTimestamp	2159
6.434.2.93	setTransactionId	2159
6.434.2.94	setType	2159
6.434.2.95	setUserID	2159
6.434.2.96	toString	2159
6.434.2.97	visit	2160
6.434.3	Field Documentation	2161
6.434.3.1	arrival	2161
6.434.3.2	brokerInTime	2161
6.434.3.3	brokerOutTime	2161
6.434.3.4	brokerPath	2161
6.434.3.5	cluster	2161
6.434.3.6	compressed	2161
6.434.3.7	content	2161
6.434.3.8	correlationId	2161
6.434.3.9	dataStructure	2161
6.434.3.10	DEFAULT_MESSAGE_SIZE	2161
6.434.3.11	destination	2161
6.434.3.12	droppable	2161
6.434.3.13	expiration	2161
6.434.3.14	groupId	2161
6.434.3.15	groupSequence	2161
6.434.3.16	ID_MESSAGE	2161
6.434.3.17	marshalledProperties	2161
6.434.3.18	messageId	2161
6.434.3.19	originalDestination	2161
6.434.3.20	originalTransactionId	2161
6.434.3.21	persistent	2161
6.434.3.22	priority	2161
6.434.3.23	producerId	2161
6.434.3.24	recievedByDFBridge	2161
6.434.3.25	redeliveryCounter	2161
6.434.3.26	replyTo	2161
6.434.3.27	targetConsumerId	2161

6.434.3.28	timestamp	2161
6.434.3.29	ransactionId	2161
6.434.3.30	type	2161
6.434.3.31	userId	2161
6.435	cms::Message Class Reference	2163
6.435.1	Detailed Description	2166
6.435.2	Constructor & Destructor Documentation	2167
6.435.2.1	~Message	2167
6.435.3	Member Function Documentation	2167
6.435.3.1	acknowledge	2167
6.435.3.2	clearBody	2168
6.435.3.3	clearProperties	2168
6.435.3.4	clone	2168
6.435.3.5	getBooleanProperty	2169
6.435.3.6	getByteProperty	2169
6.435.3.7	getCMSCorrelationID	2170
6.435.3.8	getCMSDeliveryMode	2170
6.435.3.9	getCMSDestination	2170
6.435.3.10	getCMSExpiration	2171
6.435.3.11	getCMSMessageID	2171
6.435.3.12	getCMSPriority	2172
6.435.3.13	getCMSRedelivered	2173
6.435.3.14	getCMSReplyTo	2173
6.435.3.15	getCMSTimestamp	2173
6.435.3.16	getCMSType	2174
6.435.3.17	getDoubleProperty	2174
6.435.3.18	getFloatProperty	2175
6.435.3.19	getIntProperty	2175
6.435.3.20	getLongProperty	2176
6.435.3.21	getPropertyNames	2176
6.435.3.22	getShortProperty	2177
6.435.3.23	getStringProperty	2177
6.435.3.24	propertyExists	2178
6.435.3.25	setBooleanProperty	2178
6.435.3.26	setByteProperty	2179
6.435.3.27	setCMSCorrelationID	2179

6.435.3.28	setCMSDeliveryMode	2180
6.435.3.29	setCMSDestination	2180
6.435.3.30	setCMSExpiration	2181
6.435.3.31	setCMSMessageID	2181
6.435.3.32	setCMSPriority	2181
6.435.3.33	setCMSRedelivered	2182
6.435.3.34	setCMSReplyTo	2182
6.435.3.35	setCMSTimestamp	2183
6.435.3.36	setCMSType	2183
6.435.3.37	setDoubleProperty	2184
6.435.3.38	setFloatProperty	2184
6.435.3.39	setIntProperty	2185
6.435.3.40	setLongProperty	2185
6.435.3.41	setShortProperty	2186
6.435.3.42	setStringProperty	2186
6.436	activemq::commands::MessageAck Class Reference	2188
6.436.1	Constructor & Destructor Documentation	2189
6.436.1.1	MessageAck	2189
6.436.1.2	MessageAck	2189
6.436.1.3	~MessageAck	2189
6.436.2	Member Function Documentation	2189
6.436.2.1	cloneDataStructure	2189
6.436.2.2	copyDataStructure	2189
6.436.2.3	equals	2190
6.436.2.4	getAckType	2190
6.436.2.5	getConsumerId	2190
6.436.2.6	getConsumerId	2190
6.436.2.7	getDataStructureType	2190
6.436.2.8	getDestination	2191
6.436.2.9	getDestination	2191
6.436.2.10	getFirstMessageId	2191
6.436.2.11	getFirstMessageId	2191
6.436.2.12	getLastMessageId	2191
6.436.2.13	getLastMessageId	2191
6.436.2.14	getMessageCount	2191
6.436.2.15	getTransactionId	2191

6.436.2.16	getTransactionId	2191
6.436.2.17	isMessageAck	2191
6.436.2.18	operator=	2192
6.436.2.19	setAckType	2192
6.436.2.20	setConsumerId	2192
6.436.2.21	setDestination	2192
6.436.2.22	setFirstMessageId	2192
6.436.2.23	setLastMessageId	2192
6.436.2.24	setMessageCount	2192
6.436.2.25	setTransactionId	2192
6.436.2.26	toString	2192
6.436.2.27	visit	2192
6.436.3	Field Documentation	2193
6.436.3.1	ackType	2193
6.436.3.2	consumerId	2193
6.436.3.3	destination	2193
6.436.3.4	firstMessageId	2193
6.436.3.5	ID_MESSAGEACK	2193
6.436.3.6	lastMessageId	2193
6.436.3.7	messageCount	2193
6.436.3.8	transactionId	2193
6.437	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference	2194
6.437.1	Detailed Description	2194
6.437.2	Constructor & Destructor Documentation	2195
6.437.2.1	MessageAckMarshaller	2195
6.437.2.2	~MessageAckMarshaller	2195
6.437.3	Member Function Documentation	2195
6.437.3.1	createObject	2195
6.437.3.2	getDataStructureType	2195
6.437.3.3	looseMarshal	2195
6.437.3.4	looseUnmarshal	2196
6.437.3.5	tightMarshal1	2196
6.437.3.6	tightMarshal2	2196
6.437.3.7	tightUnmarshal	2197
6.438	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference	2198
6.438.1	Detailed Description	2198

6.438.2 Constructor & Destructor Documentation	2199
6.438.2.1 MessageAckMarshaller	2199
6.438.2.2 ~MessageAckMarshaller	2199
6.438.3 Member Function Documentation	2199
6.438.3.1 createObject	2199
6.438.3.2 getDataStructureType	2199
6.438.3.3 looseMarshal	2199
6.438.3.4 looseUnmarshal	2200
6.438.3.5 tightMarshal1	2200
6.438.3.6 tightMarshal2	2200
6.438.3.7 tightUnmarshal	2201
6.439activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference	2202
6.439.1 Detailed Description	2202
6.439.2 Constructor & Destructor Documentation	2203
6.439.2.1 MessageAckMarshaller	2203
6.439.2.2 ~MessageAckMarshaller	2203
6.439.3 Member Function Documentation	2203
6.439.3.1 createObject	2203
6.439.3.2 getDataStructureType	2203
6.439.3.3 looseMarshal	2203
6.439.3.4 looseUnmarshal	2204
6.439.3.5 tightMarshal1	2204
6.439.3.6 tightMarshal2	2204
6.439.3.7 tightUnmarshal	2205
6.440activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference	2206
6.440.1 Detailed Description	2206
6.440.2 Constructor & Destructor Documentation	2207
6.440.2.1 MessageAckMarshaller	2207
6.440.2.2 ~MessageAckMarshaller	2207
6.440.3 Member Function Documentation	2207
6.440.3.1 createObject	2207
6.440.3.2 getDataStructureType	2207
6.440.3.3 looseMarshal	2207
6.440.3.4 looseUnmarshal	2208
6.440.3.5 tightMarshal1	2208
6.440.3.6 tightMarshal2	2208

6.440.3.7 tightUnmarshal	2209
6.441activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference	2210
6.441.1 Detailed Description	2210
6.441.2 Constructor & Destructor Documentation	2211
6.441.2.1 MessageAckMarshaller	2211
6.441.2.2 ~MessageAckMarshaller	2211
6.441.3 Member Function Documentation	2211
6.441.3.1 createObject	2211
6.441.3.2 getDataStructureType	2211
6.441.3.3 looseMarshal	2211
6.441.3.4 looseUnmarshal	2212
6.441.3.5 tightMarshal1	2212
6.441.3.6 tightMarshal2	2212
6.441.3.7 tightUnmarshal	2213
6.442cms::MessageConsumer Class Reference	2214
6.442.1 Detailed Description	2214
6.442.2 Constructor & Destructor Documentation	2215
6.442.2.1 ~MessageConsumer	2215
6.442.3 Member Function Documentation	2215
6.442.3.1 getMessageListener	2215
6.442.3.2 getMessageSelector	2215
6.442.3.3 receive	2215
6.442.3.4 receive	2216
6.442.3.5 receiveNoWait	2216
6.442.3.6 setMessageListener	2216
6.443activemq::cmsutil::MessageCreator Class Reference	2218
6.443.1 Detailed Description	2218
6.443.2 Constructor & Destructor Documentation	2218
6.443.2.1 ~MessageCreator	2218
6.443.3 Member Function Documentation	2218
6.443.3.1 createMessage	2218
6.444activemq::commands::MessageDispatch Class Reference	2219
6.444.1 Constructor & Destructor Documentation	2220
6.444.1.1 MessageDispatch	2220
6.444.1.2 MessageDispatch	2220
6.444.1.3 ~MessageDispatch	2220

6.444.2 Member Function Documentation	2220
6.444.2.1 cloneDataStructure	2220
6.444.2.2 copyDataStructure	2220
6.444.2.3 equals	2221
6.444.2.4 getConsumerId	2221
6.444.2.5 getConsumerId	2221
6.444.2.6 getDataStructureType	2221
6.444.2.7 getDestination	2222
6.444.2.8 getDestination	2222
6.444.2.9 getMessage	2222
6.444.2.10getMessage	2222
6.444.2.11getRedeliveryCounter	2222
6.444.2.12sMessageDispatch	2222
6.444.2.13operator=	2222
6.444.2.14setConsumerId	2222
6.444.2.15setDestination	2222
6.444.2.16setMessage	2222
6.444.2.17setRedeliveryCounter	2222
6.444.2.18oString	2222
6.444.2.19visit	2223
6.444.3 Field Documentation	2223
6.444.3.1 consumerId	2223
6.444.3.2 destination	2223
6.444.3.3 ID_MESSAGEDISPATCH	2223
6.444.3.4 message	2223
6.444.3.5 redeliveryCounter	2223
6.445activemq::core::MessageDispatchChannel Class Reference	2224
6.445.1 Constructor & Destructor Documentation	2225
6.445.1.1 MessageDispatchChannel	2225
6.445.1.2 ~MessageDispatchChannel	2225
6.445.2 Member Function Documentation	2225
6.445.2.1 clear	2225
6.445.2.2 close	2225
6.445.2.3 dequeue	2226
6.445.2.4 dequeueNoWait	2226
6.445.2.5 enqueue	2226

6.445.2.6 enqueueFirst	2226
6.445.2.7 isClosed	2226
6.445.2.8 isEmpty	2227
6.445.2.9 isRunning	2227
6.445.2.10 lock	2227
6.445.2.11 notify	2227
6.445.2.12 notifyAll	2227
6.445.2.13 peek	2228
6.445.2.14 removeAll	2228
6.445.2.15 size	2228
6.445.2.16 start	2228
6.445.2.17 stop	2228
6.445.2.18 tryLock	2228
6.445.2.19 unlock	2229
6.445.2.20 wait	2229
6.445.2.21 wait	2229
6.445.2.22 wait	2230
6.446activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class	
Reference	2231
6.446.1 Detailed Description	2231
6.446.2 Constructor & Destructor Documentation	2232
6.446.2.1 MessageDispatchMarshaller	2232
6.446.2.2 ~MessageDispatchMarshaller	2232
6.446.3 Member Function Documentation	2232
6.446.3.1 createObject	2232
6.446.3.2 getDataStructureType	2232
6.446.3.3 looseMarshal	2232
6.446.3.4 looseUnmarshal	2233
6.446.3.5 tightMarshal1	2233
6.446.3.6 tightMarshal2	2233
6.446.3.7 tightUnmarshal	2234
6.447activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller Class	
Reference	2235
6.447.1 Detailed Description	2235
6.447.2 Constructor & Destructor Documentation	2236
6.447.2.1 MessageDispatchMarshaller	2236
6.447.2.2 ~MessageDispatchMarshaller	2236

6.447.3 Member Function Documentation	2236
6.447.3.1 createObject	2236
6.447.3.2 getDataStructureType	2236
6.447.3.3 looseMarshal	2236
6.447.3.4 looseUnmarshal	2237
6.447.3.5 tightMarshal1	2237
6.447.3.6 tightMarshal2	2237
6.447.3.7 tightUnmarshal	2238
6.448activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller Class	
Reference	2239
6.448.1 Detailed Description	2239
6.448.2 Constructor & Destructor Documentation	2240
6.448.2.1 MessageDispatchMarshaller	2240
6.448.2.2 ~MessageDispatchMarshaller	2240
6.448.3 Member Function Documentation	2240
6.448.3.1 createObject	2240
6.448.3.2 getDataStructureType	2240
6.448.3.3 looseMarshal	2240
6.448.3.4 looseUnmarshal	2241
6.448.3.5 tightMarshal1	2241
6.448.3.6 tightMarshal2	2241
6.448.3.7 tightUnmarshal	2242
6.449activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class	
Reference	2243
6.449.1 Detailed Description	2243
6.449.2 Constructor & Destructor Documentation	2244
6.449.2.1 MessageDispatchMarshaller	2244
6.449.2.2 ~MessageDispatchMarshaller	2244
6.449.3 Member Function Documentation	2244
6.449.3.1 createObject	2244
6.449.3.2 getDataStructureType	2244
6.449.3.3 looseMarshal	2244
6.449.3.4 looseUnmarshal	2245
6.449.3.5 tightMarshal1	2245
6.449.3.6 tightMarshal2	2245
6.449.3.7 tightUnmarshal	2246

6.450	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	Class	
	Reference		2247
6.450.1	Detailed Description		2247
6.450.2	Constructor & Destructor Documentation		2248
6.450.2.1	MessageDispatchMarshaller		2248
6.450.2.2	~MessageDispatchMarshaller		2248
6.450.3	Member Function Documentation		2248
6.450.3.1	createObject		2248
6.450.3.2	getDataStructureType		2248
6.450.3.3	looseMarshal		2248
6.450.3.4	looseUnmarshal		2249
6.450.3.5	tightMarshal1		2249
6.450.3.6	tightMarshal2		2249
6.450.3.7	tightUnmarshal		2250
6.451	activemq::commands::MessageDispatchNotification	Class Reference	2251
6.451.1	Constructor & Destructor Documentation		2252
6.451.1.1	MessageDispatchNotification		2252
6.451.1.2	MessageDispatchNotification		2252
6.451.1.3	~MessageDispatchNotification		2252
6.451.2	Member Function Documentation		2252
6.451.2.1	cloneDataStructure		2252
6.451.2.2	copyDataStructure		2252
6.451.2.3	equals		2253
6.451.2.4	getConsumerId		2253
6.451.2.5	getConsumerId		2253
6.451.2.6	getDataStructureType		2253
6.451.2.7	getDeliverySequenceId		2254
6.451.2.8	getDestination		2254
6.451.2.9	getDestination		2254
6.451.2.10	getMessageId		2254
6.451.2.11	getMessageId		2254
6.451.2.12	setMessageDispatchNotification		2254
6.451.2.13	operator=		2255
6.451.2.14	setConsumerId		2255
6.451.2.15	setDeliverySequenceId		2255
6.451.2.16	setDestination		2255
6.451.2.17	setMessageId		2255

6.451.2.18	oString	2255
6.451.2.19	visit	2255
6.451.3	Field Documentation	2256
6.451.3.1	consumerId	2256
6.451.3.2	deliverySequenceId	2256
6.451.3.3	destination	2256
6.451.3.4	ID_MESSAGE_DISPATCH_NOTIFICATION	2256
6.451.3.5	messageId	2256
6.452	activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	
	Class Reference	2257
6.452.1	Detailed Description	2257
6.452.2	Constructor & Destructor Documentation	2258
6.452.2.1	MessageDispatchNotificationMarshaller	2258
6.452.2.2	~MessageDispatchNotificationMarshaller	2258
6.452.3	Member Function Documentation	2258
6.452.3.1	createObject	2258
6.452.3.2	getDataStructureType	2258
6.452.3.3	looseMarshal	2258
6.452.3.4	looseUnmarshal	2259
6.452.3.5	tightMarshal1	2259
6.452.3.6	tightMarshal2	2259
6.452.3.7	tightUnmarshal	2260
6.453	activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	
	Class Reference	2261
6.453.1	Detailed Description	2261
6.453.2	Constructor & Destructor Documentation	2262
6.453.2.1	MessageDispatchNotificationMarshaller	2262
6.453.2.2	~MessageDispatchNotificationMarshaller	2262
6.453.3	Member Function Documentation	2262
6.453.3.1	createObject	2262
6.453.3.2	getDataStructureType	2262
6.453.3.3	looseMarshal	2262
6.453.3.4	looseUnmarshal	2263
6.453.3.5	tightMarshal1	2263
6.453.3.6	tightMarshal2	2263
6.453.3.7	tightUnmarshal	2264

6.454	activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
	Class Reference	2265
6.454.1	Detailed Description	2265
6.454.2	Constructor & Destructor Documentation	2266
	6.454.2.1 MessageDispatchNotificationMarshaller	2266
	6.454.2.2 ~MessageDispatchNotificationMarshaller	2266
6.454.3	Member Function Documentation	2266
	6.454.3.1 createObject	2266
	6.454.3.2 getDataStructureType	2266
	6.454.3.3 looseMarshal	2266
	6.454.3.4 looseUnmarshal	2267
	6.454.3.5 tightMarshal1	2267
	6.454.3.6 tightMarshal2	2267
	6.454.3.7 tightUnmarshal	2268
6.455	activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	
	Class Reference	2269
6.455.1	Detailed Description	2269
6.455.2	Constructor & Destructor Documentation	2270
	6.455.2.1 MessageDispatchNotificationMarshaller	2270
	6.455.2.2 ~MessageDispatchNotificationMarshaller	2270
6.455.3	Member Function Documentation	2270
	6.455.3.1 createObject	2270
	6.455.3.2 getDataStructureType	2270
	6.455.3.3 looseMarshal	2270
	6.455.3.4 looseUnmarshal	2271
	6.455.3.5 tightMarshal1	2271
	6.455.3.6 tightMarshal2	2271
	6.455.3.7 tightUnmarshal	2272
6.456	activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	
	Class Reference	2273
6.456.1	Detailed Description	2273
6.456.2	Constructor & Destructor Documentation	2274
	6.456.2.1 MessageDispatchNotificationMarshaller	2274
	6.456.2.2 ~MessageDispatchNotificationMarshaller	2274
6.456.3	Member Function Documentation	2274
	6.456.3.1 createObject	2274
	6.456.3.2 getDataStructureType	2274

6.456.3.3 looseMarshal	2274
6.456.3.4 looseUnmarshal	2275
6.456.3.5 tightMarshal1	2275
6.456.3.6 tightMarshal2	2275
6.456.3.7 tightUnmarshal	2276
6.457cms::MessageEOFException Class Reference	2277
6.457.1 Detailed Description	2277
6.457.2 Constructor & Destructor Documentation	2277
6.457.2.1 MessageEOFException	2277
6.457.2.2 MessageEOFException	2277
6.457.2.3 MessageEOFException	2277
6.457.2.4 MessageEOFException	2277
6.457.2.5 ~MessageEOFException	2277
6.458cms::MessageFormatException Class Reference	2278
6.458.1 Detailed Description	2278
6.458.2 Constructor & Destructor Documentation	2278
6.458.2.1 MessageFormatException	2278
6.458.2.2 MessageFormatException	2278
6.458.2.3 MessageFormatException	2278
6.458.2.4 MessageFormatException	2278
6.458.2.5 ~MessageFormatException	2278
6.459activemq::commands::MessageId Class Reference	2279
6.459.1 Member Typedef Documentation	2280
6.459.1.1 COMPARATOR	2280
6.459.2 Constructor & Destructor Documentation	2280
6.459.2.1 MessageId	2280
6.459.2.2 MessageId	2280
6.459.2.3 ~MessageId	2280
6.459.3 Member Function Documentation	2280
6.459.3.1 cloneDataStructure	2280
6.459.3.2 compareTo	2280
6.459.3.3 copyDataStructure	2280
6.459.3.4 equals	2281
6.459.3.5 equals	2281
6.459.3.6 getBrokerSequenceId	2281
6.459.3.7 getDataStructureType	2281

6.459.3.8	getProducerId	2282
6.459.3.9	getProducerId	2282
6.459.3.10	getProducerSequenceId	2282
6.459.3.11	operator<	2282
6.459.3.12	operator=	2282
6.459.3.13	operator==	2282
6.459.3.14	setBrokerSequenceId	2282
6.459.3.15	setProducerId	2282
6.459.3.16	setProducerSequenceId	2282
6.459.3.17	toString	2282
6.459.4	Field Documentation	2283
6.459.4.1	brokerSequenceId	2283
6.459.4.2	ID_MESSAGEID	2283
6.459.4.3	producerId	2283
6.459.4.4	producerSequenceId	2283
6.460	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference	2284
6.460.1	Detailed Description	2284
6.460.2	Constructor & Destructor Documentation	2285
6.460.2.1	MessageIdMarshaller	2285
6.460.2.2	~MessageIdMarshaller	2285
6.460.3	Member Function Documentation	2285
6.460.3.1	createObject	2285
6.460.3.2	getDataStructureType	2285
6.460.3.3	looseMarshal	2285
6.460.3.4	looseUnmarshal	2286
6.460.3.5	tightMarshal1	2286
6.460.3.6	tightMarshal2	2286
6.460.3.7	tightUnmarshal	2287
6.461	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference	2288
6.461.1	Detailed Description	2288
6.461.2	Constructor & Destructor Documentation	2289
6.461.2.1	MessageIdMarshaller	2289
6.461.2.2	~MessageIdMarshaller	2289
6.461.3	Member Function Documentation	2289
6.461.3.1	createObject	2289
6.461.3.2	getDataStructureType	2289

6.461.3.3 looseMarshal	2289
6.461.3.4 looseUnmarshal	2290
6.461.3.5 tightMarshal1	2290
6.461.3.6 tightMarshal2	2290
6.461.3.7 tightUnmarshal	2291
6.462activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference	2292
6.462.1 Detailed Description	2292
6.462.2 Constructor & Destructor Documentation	2293
6.462.2.1 MessageIdMarshaller	2293
6.462.2.2 ~MessageIdMarshaller	2293
6.462.3 Member Function Documentation	2293
6.462.3.1 createObject	2293
6.462.3.2 getDataStructureType	2293
6.462.3.3 looseMarshal	2293
6.462.3.4 looseUnmarshal	2294
6.462.3.5 tightMarshal1	2294
6.462.3.6 tightMarshal2	2294
6.462.3.7 tightUnmarshal	2295
6.463activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference	2296
6.463.1 Detailed Description	2296
6.463.2 Constructor & Destructor Documentation	2297
6.463.2.1 MessageIdMarshaller	2297
6.463.2.2 ~MessageIdMarshaller	2297
6.463.3 Member Function Documentation	2297
6.463.3.1 createObject	2297
6.463.3.2 getDataStructureType	2297
6.463.3.3 looseMarshal	2297
6.463.3.4 looseUnmarshal	2298
6.463.3.5 tightMarshal1	2298
6.463.3.6 tightMarshal2	2298
6.463.3.7 tightUnmarshal	2299
6.464activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference	2300
6.464.1 Detailed Description	2300
6.464.2 Constructor & Destructor Documentation	2301
6.464.2.1 MessageIdMarshaller	2301
6.464.2.2 ~MessageIdMarshaller	2301

6.464.3 Member Function Documentation	2301
6.464.3.1 createObject	2301
6.464.3.2 getDataStructureType	2301
6.464.3.3 looseMarshal	2301
6.464.3.4 looseUnmarshal	2302
6.464.3.5 tightMarshal1	2302
6.464.3.6 tightMarshal2	2302
6.464.3.7 tightUnmarshal	2303
6.465 cms::MessageListener Class Reference	2304
6.465.1 Detailed Description	2304
6.465.2 Constructor & Destructor Documentation	2304
6.465.2.1 ~MessageListener	2304
6.465.3 Member Function Documentation	2304
6.465.3.1 onMessage	2304
6.466 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference	2305
6.466.1 Detailed Description	2305
6.466.2 Constructor & Destructor Documentation	2306
6.466.2.1 MessageMarshaller	2306
6.466.2.2 ~MessageMarshaller	2306
6.466.3 Member Function Documentation	2306
6.466.3.1 looseMarshal	2306
6.466.3.2 looseUnmarshal	2306
6.466.3.3 tightMarshal1	2307
6.466.3.4 tightMarshal2	2308
6.466.3.5 tightUnmarshal	2308
6.467 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference	2310
6.467.1 Detailed Description	2310
6.467.2 Constructor & Destructor Documentation	2311
6.467.2.1 MessageMarshaller	2311
6.467.2.2 ~MessageMarshaller	2311
6.467.3 Member Function Documentation	2311
6.467.3.1 looseMarshal	2311
6.467.3.2 looseUnmarshal	2311
6.467.3.3 tightMarshal1	2312
6.467.3.4 tightMarshal2	2313
6.467.3.5 tightUnmarshal	2313

6.468	activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference	2315
6.468.1	Detailed Description	2315
6.468.2	Constructor & Destructor Documentation	2316
6.468.2.1	MessageMarshaller	2316
6.468.2.2	~MessageMarshaller	2316
6.468.3	Member Function Documentation	2316
6.468.3.1	looseMarshal	2316
6.468.3.2	looseUnmarshal	2316
6.468.3.3	tightMarshal1	2317
6.468.3.4	tightMarshal2	2318
6.468.3.5	tightUnmarshal	2318
6.469	activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference	2320
6.469.1	Detailed Description	2320
6.469.2	Constructor & Destructor Documentation	2321
6.469.2.1	MessageMarshaller	2321
6.469.2.2	~MessageMarshaller	2321
6.469.3	Member Function Documentation	2321
6.469.3.1	looseMarshal	2321
6.469.3.2	looseUnmarshal	2321
6.469.3.3	tightMarshal1	2322
6.469.3.4	tightMarshal2	2323
6.469.3.5	tightUnmarshal	2323
6.470	activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference	2325
6.470.1	Detailed Description	2325
6.470.2	Constructor & Destructor Documentation	2326
6.470.2.1	MessageMarshaller	2326
6.470.2.2	~MessageMarshaller	2326
6.470.3	Member Function Documentation	2326
6.470.3.1	looseMarshal	2326
6.470.3.2	looseUnmarshal	2326
6.470.3.3	tightMarshal1	2327
6.470.3.4	tightMarshal2	2328
6.470.3.5	tightUnmarshal	2328
6.471	cms::MessageNotReadableException Class Reference	2330
6.471.1	Detailed Description	2330
6.471.2	Constructor & Destructor Documentation	2330

6.471.2.1 MessageNotReadableException	2330
6.471.2.2 MessageNotReadableException	2330
6.471.2.3 MessageNotReadableException	2330
6.471.2.4 MessageNotReadableException	2330
6.471.2.5 ~MessageNotReadableException	2330
6.472cms::MessageNotWriteableException Class Reference	2331
6.472.1 Detailed Description	2331
6.472.2 Constructor & Destructor Documentation	2331
6.472.2.1 MessageNotWriteableException	2331
6.472.2.2 MessageNotWriteableException	2331
6.472.2.3 MessageNotWriteableException	2331
6.472.2.4 MessageNotWriteableException	2331
6.472.2.5 ~MessageNotWriteableException	2331
6.473cms::MessageProducer Class Reference	2332
6.473.1 Detailed Description	2333
6.473.2 Constructor & Destructor Documentation	2333
6.473.2.1 ~MessageProducer	2333
6.473.3 Member Function Documentation	2333
6.473.3.1 getDeliveryMode	2333
6.473.3.2 getDisableMessageID	2334
6.473.3.3 getDisableMessageTimeStamp	2334
6.473.3.4 getPriority	2334
6.473.3.5 getTimeToLive	2335
6.473.3.6 send	2335
6.473.3.7 send	2335
6.473.3.8 send	2336
6.473.3.9 send	2336
6.473.3.10 setDeliveryMode	2337
6.473.3.11 setDisableMessageID	2337
6.473.3.12 setDisableMessageTimeStamp	2337
6.473.3.13 setPriority	2338
6.473.3.14 setTimeToLive	2338
6.474activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference	2339
6.474.1 Detailed Description	2340
6.474.2 Constructor & Destructor Documentation	2340
6.474.2.1 MessagePropertyInterceptor	2340

6.474.2.2	~MessagePropertyInterceptor	2341
6.474.3	Member Function Documentation	2341
6.474.3.1	getBooleanProperty	2341
6.474.3.2	getByteProperty	2341
6.474.3.3	getDoubleProperty	2341
6.474.3.4	getFloatProperty	2341
6.474.3.5	getIntProperty	2342
6.474.3.6	getLongProperty	2342
6.474.3.7	getShortProperty	2342
6.474.3.8	getStringProperty	2343
6.474.3.9	setBooleanProperty	2343
6.474.3.10	setByteProperty	2343
6.474.3.11	setDoubleProperty	2343
6.474.3.12	setFloatProperty	2344
6.474.3.13	setIntProperty	2344
6.474.3.14	setLongProperty	2344
6.474.3.15	setShortProperty	2344
6.474.3.16	setStringProperty	2344
6.475	activemq::commands::MessagePull Class Reference	2346
6.475.1	Constructor & Destructor Documentation	2347
6.475.1.1	MessagePull	2347
6.475.1.2	MessagePull	2347
6.475.1.3	~MessagePull	2347
6.475.2	Member Function Documentation	2347
6.475.2.1	cloneDataStructure	2347
6.475.2.2	copyDataStructure	2347
6.475.2.3	equals	2348
6.475.2.4	getConsumerId	2348
6.475.2.5	getConsumerId	2348
6.475.2.6	getCorrelationId	2348
6.475.2.7	getCorrelationId	2348
6.475.2.8	getDataStructureType	2348
6.475.2.9	getDestination	2349
6.475.2.10	getDestination	2349
6.475.2.11	getMessageId	2349
6.475.2.12	getMessageId	2349

6.475.2.13	getTimeout	2349
6.475.2.14	operator=	2349
6.475.2.15	setConsumerId	2349
6.475.2.16	setCorrelationId	2349
6.475.2.17	setDestination	2349
6.475.2.18	setMessageId	2349
6.475.2.19	setTimeout	2349
6.475.2.20	toString	2349
6.475.2.21	visit	2349
6.475.3	Field Documentation	2350
6.475.3.1	consumerId	2350
6.475.3.2	correlationId	2350
6.475.3.3	destination	2350
6.475.3.4	ID_MESSAGEPULL	2350
6.475.3.5	messageId	2350
6.475.3.6	timeout	2350
6.476	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference	2351
6.476.1	Detailed Description	2351
6.476.2	Constructor & Destructor Documentation	2352
6.476.2.1	MessagePullMarshaller	2352
6.476.2.2	~MessagePullMarshaller	2352
6.476.3	Member Function Documentation	2352
6.476.3.1	createObject	2352
6.476.3.2	getDataStructureType	2352
6.476.3.3	looseMarshal	2352
6.476.3.4	looseUnmarshal	2353
6.476.3.5	tightMarshal1	2353
6.476.3.6	tightMarshal2	2353
6.476.3.7	tightUnmarshal	2354
6.477	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference	2355
6.477.1	Detailed Description	2355
6.477.2	Constructor & Destructor Documentation	2356
6.477.2.1	MessagePullMarshaller	2356
6.477.2.2	~MessagePullMarshaller	2356
6.477.3	Member Function Documentation	2356
6.477.3.1	createObject	2356

6.477.3.2	getDataStructureType	2356
6.477.3.3	looseMarshal	2356
6.477.3.4	looseUnmarshal	2357
6.477.3.5	tightMarshal1	2357
6.477.3.6	tightMarshal2	2357
6.477.3.7	tightUnmarshal	2358
6.478	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference	2359
6.478.1	Detailed Description	2359
6.478.2	Constructor & Destructor Documentation	2360
6.478.2.1	MessagePullMarshaller	2360
6.478.2.2	~MessagePullMarshaller	2360
6.478.3	Member Function Documentation	2360
6.478.3.1	createObject	2360
6.478.3.2	getDataStructureType	2360
6.478.3.3	looseMarshal	2360
6.478.3.4	looseUnmarshal	2361
6.478.3.5	tightMarshal1	2361
6.478.3.6	tightMarshal2	2361
6.478.3.7	tightUnmarshal	2362
6.479	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference	2363
6.479.1	Detailed Description	2363
6.479.2	Constructor & Destructor Documentation	2364
6.479.2.1	MessagePullMarshaller	2364
6.479.2.2	~MessagePullMarshaller	2364
6.479.3	Member Function Documentation	2364
6.479.3.1	createObject	2364
6.479.3.2	getDataStructureType	2364
6.479.3.3	looseMarshal	2364
6.479.3.4	looseUnmarshal	2365
6.479.3.5	tightMarshal1	2365
6.479.3.6	tightMarshal2	2365
6.479.3.7	tightUnmarshal	2366
6.480	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference	2367
6.480.1	Detailed Description	2367
6.480.2	Constructor & Destructor Documentation	2368
6.480.2.1	MessagePullMarshaller	2368

6.480.2.2	~MessagePullMarshaller	2368
6.480.3	Member Function Documentation	2368
6.480.3.1	createObject	2368
6.480.3.2	getDataStructureType	2368
6.480.3.3	looseMarshal	2368
6.480.3.4	looseUnmarshal	2369
6.480.3.5	tightMarshal1	2369
6.480.3.6	tightMarshal2	2369
6.480.3.7	tightUnmarshal	2370
6.481	activemq::transport::mock::MockTransport Class Reference	2371
6.481.1	Detailed Description	2373
6.481.2	Constructor & Destructor Documentation	2373
6.481.2.1	MockTransport	2373
6.481.2.2	~MockTransport	2373
6.481.3	Member Function Documentation	2373
6.481.3.1	close	2373
6.481.3.2	fireCommand	2374
6.481.3.3	fireException	2374
6.481.3.4	getInstance	2374
6.481.3.5	getNumReceivedMessageBeforeFail	2374
6.481.3.6	getNumReceivedMessages	2374
6.481.3.7	getNumSentKeepAlives	2374
6.481.3.8	getNumSentKeepAlivesBeforeFail	2374
6.481.3.9	getNumSentMessageBeforeFail	2374
6.481.3.10	getNumSentMessages	2374
6.481.3.11	getRemoteAddress	2374
6.481.3.12	getTransportListener	2375
6.481.3.13	getWireFormat	2375
6.481.3.14	isClosed	2375
6.481.3.15	isConnected	2375
6.481.3.16	isFailOnClose	2376
6.481.3.17	isFailOnKeepAliveSends	2376
6.481.3.18	isFailOnReceiveMessage	2376
6.481.3.19	isFailOnSendMessage	2376
6.481.3.20	isFailOnStart	2376
6.481.3.21	isFailOnStop	2376

6.481.3.22	isFaultTolerant	2376
6.481.3.23	narrow	2376
6.481.3.24	neway	2376
6.481.3.25	reconnect	2377
6.481.3.26	request	2377
6.481.3.27	request	2377
6.481.3.28	setFailOnClose	2379
6.481.3.29	setFailOnKeepAliveSends	2379
6.481.3.30	setFailOnReceiveMessage	2379
6.481.3.31	setFailOnSendMessage	2379
6.481.3.32	setFailOnStart	2379
6.481.3.33	setFailOnStop	2379
6.481.3.34	setNumReceivedMessageBeforeFail	2379
6.481.3.35	setNumReceivedMessages	2379
6.481.3.36	setNumSentKeepAlives	2379
6.481.3.37	setNumSentKeepAlivesBeforeFail	2379
6.481.3.38	setNumSentMessageBeforeFail	2379
6.481.3.39	setNumSentMessages	2379
6.481.3.40	setOutgoingListener	2379
6.481.3.41	setResponseBuilder	2380
6.481.3.42	setTransportListener	2380
6.481.3.43	setWireFormat	2380
6.481.3.44	start	2380
6.481.3.45	stop	2380
6.482	activemq::transport::mock::MockTransportFactory Class Reference	2382
6.482.1	Detailed Description	2382
6.482.2	Constructor & Destructor Documentation	2383
6.482.2.1	~MockTransportFactory	2383
6.482.3	Member Function Documentation	2383
6.482.3.1	create	2383
6.482.3.2	createComposite	2383
6.482.3.3	doCreateComposite	2383
6.483	decaf::util::concurrent::Mutex Class Reference	2385
6.483.1	Detailed Description	2386
6.483.2	Constructor & Destructor Documentation	2386
6.483.2.1	Mutex	2386

6.483.2.2 ~Mutex	2386
6.483.3 Member Function Documentation	2386
6.483.3.1 lock	2386
6.483.3.2 notify	2386
6.483.3.3 notifyAll	2387
6.483.3.4 tryLock	2387
6.483.3.5 unlock	2387
6.483.3.6 wait	2388
6.483.3.7 wait	2388
6.483.3.8 wait	2389
6.484decaf::util::concurrent::MutexHandle Class Reference	2390
6.484.1 Constructor & Destructor Documentation	2390
6.484.1.1 MutexHandle	2390
6.484.1.2 ~MutexHandle	2390
6.484.1.3 MutexHandle	2390
6.484.1.4 ~MutexHandle	2390
6.484.2 Field Documentation	2390
6.484.2.1 lock_count	2390
6.484.2.2 lock_owner	2390
6.484.2.3 mutex	2390
6.484.2.4 mutex	2390
6.485decaf::internal::util::concurrent::MutexImpl Class Reference	2391
6.485.1 Member Function Documentation	2391
6.485.1.1 create	2391
6.485.1.2 destroy	2391
6.485.1.3 lock	2392
6.485.1.4 trylock	2392
6.485.1.5 unlock	2392
6.486activemq::commands::NetworkBridgeFilter Class Reference	2393
6.486.1 Constructor & Destructor Documentation	2394
6.486.1.1 NetworkBridgeFilter	2394
6.486.1.2 NetworkBridgeFilter	2394
6.486.1.3 ~NetworkBridgeFilter	2394
6.486.2 Member Function Documentation	2394
6.486.2.1 cloneDataStructure	2394
6.486.2.2 copyDataStructure	2394

6.486.2.3 equals	2394
6.486.2.4 getDataStructureType	2395
6.486.2.5 getNetworkBrokerId	2395
6.486.2.6 getNetworkBrokerId	2395
6.486.2.7 getNetworkTTL	2395
6.486.2.8 operator=	2395
6.486.2.9 setNetworkBrokerId	2395
6.486.2.10 setNetworkTTL	2395
6.486.2.11 toString	2395
6.486.3 Field Documentation	2396
6.486.3.1 ID_NETWORKBRIDGEFILTER	2396
6.486.3.2 networkBrokerId	2396
6.486.3.3 networkTTL	2396
6.487activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference	2397
6.487.1 Detailed Description	2397
6.487.2 Constructor & Destructor Documentation	2398
6.487.2.1 NetworkBridgeFilterMarshaller	2398
6.487.2.2 ~NetworkBridgeFilterMarshaller	2398
6.487.3 Member Function Documentation	2398
6.487.3.1 createObject	2398
6.487.3.2 getDataStructureType	2398
6.487.3.3 looseMarshal	2398
6.487.3.4 looseUnmarshal	2399
6.487.3.5 tightMarshal1	2399
6.487.3.6 tightMarshal2	2399
6.487.3.7 tightUnmarshal	2400
6.488activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class Reference	2401
6.488.1 Detailed Description	2401
6.488.2 Constructor & Destructor Documentation	2402
6.488.2.1 NetworkBridgeFilterMarshaller	2402
6.488.2.2 ~NetworkBridgeFilterMarshaller	2402
6.488.3 Member Function Documentation	2402
6.488.3.1 createObject	2402
6.488.3.2 getDataStructureType	2402
6.488.3.3 looseMarshal	2402

6.488.3.4 looseUnmarshal	2403
6.488.3.5 tightMarshal1	2403
6.488.3.6 tightMarshal2	2403
6.488.3.7 tightUnmarshal	2404
6.489activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class	
Reference	2405
6.489.1 Detailed Description	2405
6.489.2 Constructor & Destructor Documentation	2406
6.489.2.1 NetworkBridgeFilterMarshaller	2406
6.489.2.2 ~NetworkBridgeFilterMarshaller	2406
6.489.3 Member Function Documentation	2406
6.489.3.1 createObject	2406
6.489.3.2 getDataStructureType	2406
6.489.3.3 looseMarshal	2406
6.489.3.4 looseUnmarshal	2407
6.489.3.5 tightMarshal1	2407
6.489.3.6 tightMarshal2	2407
6.489.3.7 tightUnmarshal	2408
6.490activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class	
Reference	2409
6.490.1 Detailed Description	2409
6.490.2 Constructor & Destructor Documentation	2410
6.490.2.1 NetworkBridgeFilterMarshaller	2410
6.490.2.2 ~NetworkBridgeFilterMarshaller	2410
6.490.3 Member Function Documentation	2410
6.490.3.1 createObject	2410
6.490.3.2 getDataStructureType	2410
6.490.3.3 looseMarshal	2410
6.490.3.4 looseUnmarshal	2411
6.490.3.5 tightMarshal1	2411
6.490.3.6 tightMarshal2	2411
6.490.3.7 tightUnmarshal	2412
6.491activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class	
Reference	2413
6.491.1 Detailed Description	2413
6.491.2 Constructor & Destructor Documentation	2414
6.491.2.1 NetworkBridgeFilterMarshaller	2414

6.491.2.2 ~NetworkBridgeFilterMarshaller	2414
6.491.3 Member Function Documentation	2414
6.491.3.1 createObject	2414
6.491.3.2 getDataStructureType	2414
6.491.3.3 looseMarshal	2414
6.491.3.4 looseUnmarshal	2415
6.491.3.5 tightMarshal1	2415
6.491.3.6 tightMarshal2	2415
6.491.3.7 tightUnmarshal	2416
6.492decaf::net::NoRouteToHostException Class Reference	2417
6.492.1 Constructor & Destructor Documentation	2417
6.492.1.1 NoRouteToHostException	2417
6.492.1.2 NoRouteToHostException	2417
6.492.1.3 NoRouteToHostException	2418
6.492.1.4 NoRouteToHostException	2418
6.492.1.5 NoRouteToHostException	2418
6.492.1.6 NoRouteToHostException	2418
6.492.1.7 ~NoRouteToHostException	2419
6.492.2 Member Function Documentation	2419
6.492.2.1 clone	2419
6.493decaf::security::NoSuchAlgorithmException Class Reference	2420
6.493.1 Constructor & Destructor Documentation	2420
6.493.1.1 NoSuchAlgorithmException	2420
6.493.1.2 NoSuchAlgorithmException	2420
6.493.1.3 NoSuchAlgorithmException	2421
6.493.1.4 NoSuchAlgorithmException	2421
6.493.1.5 NoSuchAlgorithmException	2421
6.493.1.6 NoSuchAlgorithmException	2421
6.493.1.7 ~NoSuchAlgorithmException	2422
6.493.2 Member Function Documentation	2422
6.493.2.1 clone	2422
6.494decaf::lang::exceptions::NoSuchElementException Class Reference	2423
6.494.1 Constructor & Destructor Documentation	2423
6.494.1.1 NoSuchElementException	2423
6.494.1.2 NoSuchElementException	2423
6.494.1.3 NoSuchElementException	2424

6.494.1.4 NoSuchElementException	2424
6.494.1.5 NoSuchElementException	2424
6.494.1.6 NoSuchElementException	2424
6.494.1.7 ~NoSuchElementException	2425
6.494.2 Member Function Documentation	2425
6.494.2.1 clone	2425
6.495decaf::security::NoSuchProviderException Class Reference	2426
6.495.1 Constructor & Destructor Documentation	2426
6.495.1.1 NoSuchProviderException	2426
6.495.1.2 NoSuchProviderException	2426
6.495.1.3 NoSuchProviderException	2427
6.495.1.4 NoSuchProviderException	2427
6.495.1.5 NoSuchProviderException	2427
6.495.1.6 NoSuchProviderException	2427
6.495.1.7 ~NoSuchProviderException	2428
6.495.2 Member Function Documentation	2428
6.495.2.1 clone	2428
6.496decaf::lang::exceptions::NullPointerException Class Reference	2429
6.496.1 Constructor & Destructor Documentation	2429
6.496.1.1 NullPointerException	2429
6.496.1.2 NullPointerException	2429
6.496.1.3 NullPointerException	2430
6.496.1.4 NullPointerException	2430
6.496.1.5 NullPointerException	2430
6.496.1.6 NullPointerException	2430
6.496.1.7 ~NullPointerException	2431
6.496.2 Member Function Documentation	2431
6.496.2.1 clone	2431
6.497decaf::lang::Number Class Reference	2432
6.497.1 Detailed Description	2432
6.497.2 Constructor & Destructor Documentation	2432
6.497.2.1 ~Number	2432
6.497.3 Member Function Documentation	2432
6.497.3.1 byteValue	2432
6.497.3.2 doubleValue	2433
6.497.3.3 floatValue	2433

6.497.3.4 int Value	2433
6.497.3.5 long Value	2433
6.497.3.6 short Value	2434
6.498decaf::lang::exceptions::NumberFormatException Class Reference	2435
6.498.1 Constructor & Destructor Documentation	2435
6.498.1.1 NumberFormatException	2435
6.498.1.2 NumberFormatException	2435
6.498.1.3 NumberFormatException	2436
6.498.1.4 NumberFormatException	2436
6.498.1.5 NumberFormatException	2436
6.498.1.6 NumberFormatException	2436
6.498.1.7 ~NumberFormatException	2437
6.498.2 Member Function Documentation	2437
6.498.2.1 clone	2437
6.499cms::ObjectMessage Class Reference	2438
6.499.1 Detailed Description	2438
6.499.2 Constructor & Destructor Documentation	2438
6.499.2.1 ~ObjectMessage	2438
6.500decaf::security_provider::unix::openssl::OpenSSLX500Principal Class Reference	2439
6.500.1 Detailed Description	2439
6.500.2 Constructor & Destructor Documentation	2439
6.500.2.1 OpenSSLX500Principal	2439
6.500.2.2 ~OpenSSLX500Principal	2440
6.500.3 Member Function Documentation	2440
6.500.3.1 equals	2440
6.500.3.2 getEncoded	2440
6.500.3.3 getEncoded	2440
6.500.3.4 getName	2441
6.500.3.5 getX509Name	2441
6.500.3.6 toString	2441
6.501decaf::security_provider::unix::openssl::OpenSSLX509Certificate Class Reference	2442
6.501.1 Constructor & Destructor Documentation	2443
6.501.1.1 ~OpenSSLX509Certificate	2443
6.501.2 Member Function Documentation	2443
6.501.2.1 checkValidity	2443
6.501.2.2 checkValidity	2443

6.501.2.3 equals	2443
6.501.2.4 getBasicConstraints	2444
6.501.2.5 getEncoded	2444
6.501.2.6 getIssuerUniqueID	2444
6.501.2.7 getIssuerX500Principal	2444
6.501.2.8 getKeyUsage	2444
6.501.2.9 getNotAfter	2444
6.501.2.10 getNotBefore	2444
6.501.2.11 getPublicKey	2445
6.501.2.12 getPublicKey	2445
6.501.2.13 getSigAlgName	2445
6.501.2.14 getSigAlgOID	2445
6.501.2.15 getSigAlgParams	2445
6.501.2.16 getSignature	2445
6.501.2.17 getSubjectUniqueID	2446
6.501.2.18 getSubjectX500Principal	2446
6.501.2.19 getTBSCertificate	2446
6.501.2.20 getType	2446
6.501.2.21 getVersion	2446
6.501.2.22 toString	2446
6.501.2.23 verify	2447
6.501.2.24 verify	2447
6.502activemq::wireformat::openwire::OpenWireFormat Class Reference	2448
6.502.1 Constructor & Destructor Documentation	2451
6.502.1.1 OpenWireFormat	2451
6.502.1.2 ~OpenWireFormat	2451
6.502.2 Member Function Documentation	2451
6.502.2.1 addMarshaller	2451
6.502.2.2 createNegotiator	2451
6.502.2.3 destroyMarshalers	2451
6.502.2.4 doUnmarshal	2452
6.502.2.5 getCacheSize	2452
6.502.2.6 getMaxInactivityDuration	2452
6.502.2.7 getMaxInactivityDurationInitialDelay	2452
6.502.2.8 getPreferredWireFormatInfo	2453
6.502.2.9 getVersion	2453

6.502.2.10	hasNegotiator	2453
6.502.2.11	linReceive	2453
6.502.2.12	sCacheEnabled	2453
6.502.2.13	sSizePrefixDisabled	2454
6.502.2.14	sStackTraceEnabled	2454
6.502.2.15	sTcpNoDelayEnabled	2454
6.502.2.16	sTightEncodingEnabled	2454
6.502.2.17	ooseMarshalNestedObject	2454
6.502.2.18	ooseUnmarshalNestedObject	2455
6.502.2.19	marshal	2455
6.502.2.20	renegotiateWireFormat	2455
6.502.2.21	set CacheEnabled	2456
6.502.2.22	set CacheSize	2456
6.502.2.23	set MaxInactivityDuration	2456
6.502.2.24	set MaxInactivityDurationInitialDelay	2456
6.502.2.25	set PreferredWireFormatInfo	2456
6.502.2.26	set SizePrefixDisabled	2457
6.502.2.27	set StackTraceEnabled	2457
6.502.2.28	set TcpNoDelayEnabled	2457
6.502.2.29	set TightEncodingEnabled	2457
6.502.2.30	set Version	2457
6.502.2.31	tightMarshalNestedObject1	2458
6.502.2.32	tightMarshalNestedObject2	2458
6.502.2.33	tightUnmarshalNestedObject	2458
6.502.2.34	unmarshal	2459
6.502.3	Field Documentation	2459
6.502.3.1	DEFAULT_VERSION	2459
6.502.3.2	NULL_TYPE	2459
6.503	activemq::wireformat::openwire::OpenWireFormatFactory Class Reference	2460
6.503.1	Constructor & Destructor Documentation	2460
6.503.1.1	OpenWireFormatFactory	2460
6.503.1.2	~OpenWireFormatFactory	2460
6.503.2	Member Function Documentation	2460
6.503.2.1	createWireFormat	2460
6.504	activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference	2462
6.504.1	Constructor & Destructor Documentation	2462

6.504.1.1	OpenWireFormatNegotiator	2462
6.504.1.2	~OpenWireFormatNegotiator	2463
6.504.2	Member Function Documentation	2463
6.504.2.1	close	2463
6.504.2.2	onCommand	2463
6.504.2.3	oneway	2463
6.504.2.4	onTransportException	2464
6.504.2.5	request	2464
6.504.2.6	request	2464
6.504.2.7	start	2465
6.505	activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference	2466
6.505.1	Constructor & Destructor Documentation	2466
6.505.1.1	OpenWireResponseBuilder	2466
6.505.1.2	~OpenWireResponseBuilder	2466
6.505.2	Member Function Documentation	2466
6.505.2.1	buildIncomingCommands	2466
6.505.2.2	buildResponse	2467
6.506	activemq::wireformat::openwire::utils::OpenwireStringSupport Class Reference . . .	2468
6.506.1	Constructor & Destructor Documentation	2468
6.506.1.1	OpenwireStringSupport	2468
6.506.1.2	~OpenwireStringSupport	2468
6.506.2	Member Function Documentation	2468
6.506.2.1	readString	2468
6.506.2.2	writeString	2469
6.507	decaf::io::OutputStream Class Reference	2470
6.507.1	Detailed Description	2470
6.507.2	Constructor & Destructor Documentation	2470
6.507.2.1	~OutputStream	2470
6.507.3	Member Function Documentation	2470
6.507.3.1	flush	2470
6.507.3.2	write	2471
6.507.3.3	write	2471
6.507.3.4	write	2471
6.508	activemq::commands::PartialCommand Class Reference	2473
6.508.1	Constructor & Destructor Documentation	2474
6.508.1.1	PartialCommand	2474

6.508.1.2	PartialCommand	2474
6.508.1.3	~PartialCommand	2474
6.508.2	Member Function Documentation	2474
6.508.2.1	cloneDataStructure	2474
6.508.2.2	copyDataStructure	2474
6.508.2.3	equals	2474
6.508.2.4	getCommandId	2475
6.508.2.5	getData	2475
6.508.2.6	getData	2475
6.508.2.7	getDataStructureType	2475
6.508.2.8	operator=	2475
6.508.2.9	setCommandId	2475
6.508.2.10	setData	2475
6.508.2.11	toString	2475
6.508.3	Field Documentation	2476
6.508.3.1	commandId	2476
6.508.3.2	data	2476
6.508.3.3	ID_PARTIALCOMMAND	2476
6.509	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class	
	Reference	2477
6.509.1	Detailed Description	2477
6.509.2	Constructor & Destructor Documentation	2478
6.509.2.1	PartialCommandMarshaller	2478
6.509.2.2	~PartialCommandMarshaller	2478
6.509.3	Member Function Documentation	2478
6.509.3.1	createObject	2478
6.509.3.2	getDataStructureType	2478
6.509.3.3	looseMarshal	2478
6.509.3.4	looseUnmarshal	2479
6.509.3.5	tightMarshal1	2479
6.509.3.6	tightMarshal2	2480
6.509.3.7	tightUnmarshal	2480
6.510	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class	
	Reference	2481
6.510.1	Detailed Description	2481
6.510.2	Constructor & Destructor Documentation	2482
6.510.2.1	PartialCommandMarshaller	2482

6.510.2.2 ~PartialCommandMarshaller	2482
6.510.3 Member Function Documentation	2482
6.510.3.1 createObject	2482
6.510.3.2 getDataStructureType	2482
6.510.3.3 looseMarshal	2482
6.510.3.4 looseUnmarshal	2483
6.510.3.5 tightMarshal1	2483
6.510.3.6 tightMarshal2	2484
6.510.3.7 tightUnmarshal	2484
6.511activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller Class	
Reference	2485
6.511.1 Detailed Description	2485
6.511.2 Constructor & Destructor Documentation	2486
6.511.2.1 PartialCommandMarshaller	2486
6.511.2.2 ~PartialCommandMarshaller	2486
6.511.3 Member Function Documentation	2486
6.511.3.1 createObject	2486
6.511.3.2 getDataStructureType	2486
6.511.3.3 looseMarshal	2486
6.511.3.4 looseUnmarshal	2487
6.511.3.5 tightMarshal1	2487
6.511.3.6 tightMarshal2	2488
6.511.3.7 tightUnmarshal	2488
6.512activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class	
Reference	2489
6.512.1 Detailed Description	2489
6.512.2 Constructor & Destructor Documentation	2490
6.512.2.1 PartialCommandMarshaller	2490
6.512.2.2 ~PartialCommandMarshaller	2490
6.512.3 Member Function Documentation	2490
6.512.3.1 createObject	2490
6.512.3.2 getDataStructureType	2490
6.512.3.3 looseMarshal	2490
6.512.3.4 looseUnmarshal	2491
6.512.3.5 tightMarshal1	2491
6.512.3.6 tightMarshal2	2492
6.512.3.7 tightUnmarshal	2492

6.513	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	Class	
	Reference		2493
6.513.1	Detailed Description		2493
6.513.2	Constructor & Destructor Documentation		2494
	6.513.2.1 PartialCommandMarshaller		2494
	6.513.2.2 ~PartialCommandMarshaller		2494
6.513.3	Member Function Documentation		2494
	6.513.3.1 createObject		2494
	6.513.3.2 getDataStructureType		2494
	6.513.3.3 looseMarshal		2494
	6.513.3.4 looseUnmarshal		2495
	6.513.3.5 tightMarshal1		2495
	6.513.3.6 tightMarshal2		2496
	6.513.3.7 tightUnmarshal		2496
6.514	decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference		2497
	6.514.1 Detailed Description		2498
	6.514.2 Member Typedef Documentation		2499
	6.514.2.1 CounterType		2499
	6.514.2.2 PointerType		2499
	6.514.2.3 ReferenceType		2499
	6.514.3 Constructor & Destructor Documentation		2499
	6.514.3.1 Pointer		2499
	6.514.3.2 Pointer		2499
	6.514.3.3 Pointer		2499
	6.514.3.4 Pointer		2499
	6.514.3.5 Pointer		2500
	6.514.3.6 Pointer		2500
	6.514.3.7 ~Pointer		2500
	6.514.4 Member Function Documentation		2500
	6.514.4.1 dynamicCast		2500
	6.514.4.2 get		2500
	6.514.4.3 operator!		2501
	6.514.4.4 operator!=		2501
	6.514.4.5 operator*		2501
	6.514.4.6 operator*		2501
	6.514.4.7 operator->		2501
	6.514.4.8 operator->		2501

6.514.4.9 operator=	2502
6.514.4.10 operator=	2502
6.514.4.11 operator==	2502
6.514.4.12 release	2502
6.514.4.13 reset	2502
6.514.4.14 staticCast	2503
6.514.4.15 swap	2503
6.514.5 Friends And Related Function Documentation	2503
6.514.5.1 operator!=	2503
6.514.5.2 operator!=	2503
6.514.5.3 operator==	2503
6.514.5.4 operator==	2503
6.515 decaf::lang::PointerComparator< T, R > Class Template Reference	2504
6.515.1 Detailed Description	2504
6.515.2 Member Function Documentation	2504
6.515.2.1 compare	2504
6.515.2.2 operator()	2504
6.516 activemq::cmsutil::PooledSession Class Reference	2505
6.516.1 Detailed Description	2507
6.516.2 Constructor & Destructor Documentation	2507
6.516.2.1 PooledSession	2507
6.516.2.2 ~PooledSession	2507
6.516.3 Member Function Documentation	2507
6.516.3.1 close	2507
6.516.3.2 commit	2507
6.516.3.3 createBrowser	2508
6.516.3.4 createBrowser	2508
6.516.3.5 createBytesMessage	2509
6.516.3.6 createBytesMessage	2509
6.516.3.7 createCachedConsumer	2509
6.516.3.8 createCachedProducer	2510
6.516.3.9 createConsumer	2510
6.516.3.10 createConsumer	2510
6.516.3.11 createConsumer	2511
6.516.3.12 createDurableConsumer	2511
6.516.3.13 createMapMessage	2512

6.516.3.14	createMessage	2512
6.516.3.15	createProducer	2512
6.516.3.16	createQueue	2513
6.516.3.17	createStreamMessage	2513
6.516.3.18	createTemporaryQueue	2514
6.516.3.19	createTemporaryTopic	2514
6.516.3.20	createTextMessage	2514
6.516.3.21	createTextMessage	2514
6.516.3.22	createTopic	2515
6.516.3.23	getAcknowledgeMode	2515
6.516.3.24	getSession	2515
6.516.3.25	getSession	2516
6.516.3.26	isTransacted	2516
6.516.3.27	recover	2516
6.516.3.28	rollback	2517
6.516.3.29	unsubscribe	2517
6.517	decaf::util::concurrent::PooledThread Class Reference	2518
6.517.1	Constructor & Destructor Documentation	2518
6.517.1.1	PooledThread	2518
6.517.1.2	~PooledThread	2518
6.517.2	Member Function Documentation	2518
6.517.2.1	getPooledThreadListener	2518
6.517.2.2	isBusy	2519
6.517.2.3	run	2519
6.517.2.4	setPooledThreadListener	2519
6.517.2.5	stop	2519
6.518	decaf::util::concurrent::PooledThreadListener Class Reference	2520
6.518.1	Detailed Description	2520
6.518.2	Constructor & Destructor Documentation	2520
6.518.2.1	~PooledThreadListener	2520
6.518.3	Member Function Documentation	2520
6.518.3.1	onTaskCompleted	2520
6.518.3.2	onTaskException	2521
6.518.3.3	onTaskStarted	2521
6.519	decaf::net::PortUnreachableException Class Reference	2522
6.519.1	Constructor & Destructor Documentation	2522

6.519.1.1 PortUnreachableException	2522
6.519.1.2 PortUnreachableException	2522
6.519.1.3 PortUnreachableException	2523
6.519.1.4 PortUnreachableException	2523
6.519.1.5 PortUnreachableException	2523
6.519.1.6 PortUnreachableException	2523
6.519.1.7 ~PortUnreachableException	2524
6.519.2 Member Function Documentation	2524
6.519.2.1 clone	2524
6.520activemq::util::PrimitiveList Class Reference	2525
6.520.1 Detailed Description	2527
6.520.2 Constructor & Destructor Documentation	2527
6.520.2.1 PrimitiveList	2527
6.520.2.2 ~PrimitiveList	2527
6.520.2.3 PrimitiveList	2527
6.520.2.4 PrimitiveList	2528
6.520.3 Member Function Documentation	2528
6.520.3.1 getBool	2528
6.520.3.2 getByte	2528
6.520.3.3 getByteArray	2529
6.520.3.4 getChar	2529
6.520.3.5 getDouble	2529
6.520.3.6 getFloat	2530
6.520.3.7 getInt	2530
6.520.3.8 getLong	2530
6.520.3.9 getShort	2531
6.520.3.10getString	2531
6.520.3.11setBool	2532
6.520.3.12setByte	2532
6.520.3.13setByteArray	2532
6.520.3.14setChar	2533
6.520.3.15setDouble	2533
6.520.3.16setFloat	2533
6.520.3.17setInt	2534
6.520.3.18setLong	2534
6.520.3.19setShort	2534

6.520.3.20	getString	2535
6.520.3.21	toString	2535
6.521	activemq::util::PrimitiveMap Class Reference	2536
6.521.1	Detailed Description	2538
6.521.2	Constructor & Destructor Documentation	2538
6.521.2.1	PrimitiveMap	2538
6.521.2.2	~PrimitiveMap	2538
6.521.2.3	PrimitiveMap	2538
6.521.2.4	PrimitiveMap	2538
6.521.3	Member Function Documentation	2539
6.521.3.1	getBool	2539
6.521.3.2	getByte	2539
6.521.3.3	getByteArray	2539
6.521.3.4	getChar	2540
6.521.3.5	getDouble	2540
6.521.3.6	getFloat	2541
6.521.3.7	getInt	2541
6.521.3.8	getLong	2541
6.521.3.9	getShort	2542
6.521.3.10	getString	2542
6.521.3.11	setBool	2543
6.521.3.12	setByte	2543
6.521.3.13	setByteArray	2543
6.521.3.14	setChar	2543
6.521.3.15	setDouble	2543
6.521.3.16	setFloat	2544
6.521.3.17	setInt	2544
6.521.3.18	setLong	2544
6.521.3.19	setShort	2544
6.521.3.20	getString	2545
6.521.3.21	toString	2545
6.522	activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference	2546
6.522.1	Detailed Description	2547
6.522.2	Constructor & Destructor Documentation	2547
6.522.2.1	PrimitiveTypesMarshaller	2547
6.522.2.2	~PrimitiveTypesMarshaller	2547

6.522.3 Member Function Documentation	2547
6.522.3.1 marshal	2547
6.522.3.2 marshal	2548
6.522.3.3 marshalPrimitive	2548
6.522.3.4 marshalPrimitiveList	2548
6.522.3.5 marshalPrimitiveMap	2549
6.522.3.6 unmarshal	2549
6.522.3.7 unmarshal	2549
6.522.3.8 unmarshalPrimitive	2550
6.522.3.9 unmarshalPrimitiveList	2550
6.522.3.10 unmarshalPrimitiveMap	2550
6.523 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference	2551
6.523.1 Detailed Description	2551
6.523.2 Field Documentation	2552
6.523.2.1 boolValue	2552
6.523.2.2 byteArrayValue	2552
6.523.2.3 byteValue	2552
6.523.2.4 charValue	2552
6.523.2.5 doubleValue	2552
6.523.2.6 floatValue	2552
6.523.2.7 intValue	2552
6.523.2.8 listValue	2552
6.523.2.9 longValue	2552
6.523.2.10 mapValue	2552
6.523.2.11 shortValue	2552
6.523.2.12 stringValue	2552
6.524 activemq::util::PrimitiveValueConverter Class Reference	2553
6.524.1 Detailed Description	2553
6.524.2 Constructor & Destructor Documentation	2553
6.524.2.1 PrimitiveValueConverter	2553
6.524.2.2 ~PrimitiveValueConverter	2553
6.524.3 Member Function Documentation	2553
6.524.3.1 convert	2553
6.525 activemq::util::PrimitiveValueNode Class Reference	2555
6.525.1 Detailed Description	2558
6.525.2 Member Enumeration Documentation	2558

6.525.2.1 PrimitiveType	2558
6.525.3 Constructor & Destructor Documentation	2559
6.525.3.1 PrimitiveValueNode	2559
6.525.3.2 PrimitiveValueNode	2559
6.525.3.3 PrimitiveValueNode	2559
6.525.3.4 PrimitiveValueNode	2559
6.525.3.5 PrimitiveValueNode	2560
6.525.3.6 PrimitiveValueNode	2560
6.525.3.7 PrimitiveValueNode	2560
6.525.3.8 PrimitiveValueNode	2560
6.525.3.9 PrimitiveValueNode	2560
6.525.3.10 PrimitiveValueNode	2560
6.525.3.11 PrimitiveValueNode	2561
6.525.3.12 PrimitiveValueNode	2561
6.525.3.13 PrimitiveValueNode	2561
6.525.3.14 PrimitiveValueNode	2561
6.525.3.15 PrimitiveValueNode	2561
6.525.3.16 ~PrimitiveValueNode	2561
6.525.4 Member Function Documentation	2561
6.525.4.1 clear	2561
6.525.4.2 getBool	2562
6.525.4.3 getByte	2562
6.525.4.4 getByteArray	2562
6.525.4.5 getChar	2562
6.525.4.6 getDouble	2563
6.525.4.7 getFloat	2563
6.525.4.8 getInt	2563
6.525.4.9 getList	2563
6.525.4.10 getLong	2564
6.525.4.11 getMap	2564
6.525.4.12 getShort	2564
6.525.4.13 getString	2564
6.525.4.14 getType	2565
6.525.4.15 getValue	2565
6.525.4.16 operator=	2565
6.525.4.17 operator==	2565

6.525.4.18	setBool	2565
6.525.4.19	setByte	2565
6.525.4.20	setByteArray	2566
6.525.4.21	setChar	2566
6.525.4.22	setDouble	2566
6.525.4.23	setFloat	2566
6.525.4.24	setInt	2566
6.525.4.25	setList	2566
6.525.4.26	setLong	2567
6.525.4.27	setMap	2567
6.525.4.28	setShort	2567
6.525.4.29	setString	2567
6.525.4.30	setValue	2567
6.525.4.31	toString	2568
6.526	decaf::security::Principal Class Reference	2569
6.526.1	Detailed Description	2569
6.526.2	Constructor & Destructor Documentation	2569
6.526.2.1	~Principal	2569
6.526.3	Member Function Documentation	2569
6.526.3.1	equals	2569
6.526.3.2	getName	2569
6.527	decaf::util::PriorityQueue< E > Class Template Reference	2571
6.527.1	Detailed Description	2572
6.527.2	Constructor & Destructor Documentation	2573
6.527.2.1	PriorityQueue	2573
6.527.2.2	PriorityQueue	2573
6.527.2.3	PriorityQueue	2573
6.527.2.4	PriorityQueue	2573
6.527.2.5	PriorityQueue	2574
6.527.2.6	~PriorityQueue	2574
6.527.3	Member Function Documentation	2574
6.527.3.1	add	2574
6.527.3.2	clear	2574
6.527.3.3	comparator	2575
6.527.3.4	iterator	2575
6.527.3.5	iterator	2575

6.527.3.6 offer	2575
6.527.3.7 operator=	2576
6.527.3.8 operator=	2576
6.527.3.9 peek	2576
6.527.3.10 poll	2576
6.527.3.11 remove	2577
6.527.3.12 remove	2577
6.527.3.13 size	2577
6.527.4 Friends And Related Function Documentation	2578
6.527.4.1 PriorityQueueIterator	2578
6.528 activemq::commands::ProducerAck Class Reference	2579
6.528.1 Constructor & Destructor Documentation	2580
6.528.1.1 ProducerAck	2580
6.528.1.2 ProducerAck	2580
6.528.1.3 ~ProducerAck	2580
6.528.2 Member Function Documentation	2580
6.528.2.1 cloneDataStructure	2580
6.528.2.2 copyDataStructure	2580
6.528.2.3 equals	2580
6.528.2.4 getDataStructureType	2581
6.528.2.5 getProducerId	2581
6.528.2.6 getProducerId	2581
6.528.2.7 getSize	2581
6.528.2.8 isProducerAck	2581
6.528.2.9 operator=	2581
6.528.2.10 setProducerId	2581
6.528.2.11 setSize	2581
6.528.2.12 toString	2581
6.528.2.13 visit	2582
6.528.3 Field Documentation	2582
6.528.3.1 ID_PRODUCERACK	2582
6.528.3.2 producerId	2582
6.528.3.3 size	2582
6.529 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference	2583
6.529.1 Detailed Description	2583
6.529.2 Constructor & Destructor Documentation	2584

6.529.2.1 ProducerAckMarshaller	2584
6.529.2.2 ~ProducerAckMarshaller	2584
6.529.3 Member Function Documentation	2584
6.529.3.1 createObject	2584
6.529.3.2 getDataStructureType	2584
6.529.3.3 looseMarshal	2584
6.529.3.4 looseUnmarshal	2585
6.529.3.5 tightMarshal1	2585
6.529.3.6 tightMarshal2	2585
6.529.3.7 tightUnmarshal	2586
6.530activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference	2587
6.530.1 Detailed Description	2587
6.530.2 Constructor & Destructor Documentation	2588
6.530.2.1 ProducerAckMarshaller	2588
6.530.2.2 ~ProducerAckMarshaller	2588
6.530.3 Member Function Documentation	2588
6.530.3.1 createObject	2588
6.530.3.2 getDataStructureType	2588
6.530.3.3 looseMarshal	2588
6.530.3.4 looseUnmarshal	2589
6.530.3.5 tightMarshal1	2589
6.530.3.6 tightMarshal2	2589
6.530.3.7 tightUnmarshal	2590
6.531activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference	2591
6.531.1 Detailed Description	2591
6.531.2 Constructor & Destructor Documentation	2592
6.531.2.1 ProducerAckMarshaller	2592
6.531.2.2 ~ProducerAckMarshaller	2592
6.531.3 Member Function Documentation	2592
6.531.3.1 createObject	2592
6.531.3.2 getDataStructureType	2592
6.531.3.3 looseMarshal	2592
6.531.3.4 looseUnmarshal	2593
6.531.3.5 tightMarshal1	2593
6.531.3.6 tightMarshal2	2593
6.531.3.7 tightUnmarshal	2594

6.532	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference	2595
6.532.1	Detailed Description	2595
6.532.2	Constructor & Destructor Documentation	2596
6.532.2.1	ProducerAckMarshaller	2596
6.532.2.2	~ProducerAckMarshaller	2596
6.532.3	Member Function Documentation	2596
6.532.3.1	createObject	2596
6.532.3.2	getDataStructureType	2596
6.532.3.3	looseMarshal	2596
6.532.3.4	looseUnmarshal	2597
6.532.3.5	tightMarshal1	2597
6.532.3.6	tightMarshal2	2597
6.532.3.7	tightUnmarshal	2598
6.533	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference	2599
6.533.1	Detailed Description	2599
6.533.2	Constructor & Destructor Documentation	2600
6.533.2.1	ProducerAckMarshaller	2600
6.533.2.2	~ProducerAckMarshaller	2600
6.533.3	Member Function Documentation	2600
6.533.3.1	createObject	2600
6.533.3.2	getDataStructureType	2600
6.533.3.3	looseMarshal	2600
6.533.3.4	looseUnmarshal	2601
6.533.3.5	tightMarshal1	2601
6.533.3.6	tightMarshal2	2601
6.533.3.7	tightUnmarshal	2602
6.534	activemq::cmsutil::ProducerCallback Class Reference	2603
6.534.1	Detailed Description	2603
6.534.2	Constructor & Destructor Documentation	2603
6.534.2.1	~ProducerCallback	2603
6.534.3	Member Function Documentation	2603
6.534.3.1	doInCms	2603
6.535	activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference	2604
6.535.1	Constructor & Destructor Documentation	2604
6.535.1.1	ProducerExecutor	2604
6.535.1.2	~ProducerExecutor	2604

6.535.2 Member Function Documentation	2604
6.535.2.1 doInCms	2604
6.535.2.2 getDestination	2605
6.535.3 Field Documentation	2605
6.535.3.1 action	2605
6.535.3.2 destination	2605
6.535.3.3 parent	2605
6.536activemq::commands::ProducerId Class Reference	2606
6.536.1 Member Typedef Documentation	2607
6.536.1.1 COMPARATOR	2607
6.536.2 Constructor & Destructor Documentation	2607
6.536.2.1 ProducerId	2607
6.536.2.2 ProducerId	2607
6.536.2.3 ProducerId	2607
6.536.2.4 ~ProducerId	2607
6.536.3 Member Function Documentation	2607
6.536.3.1 cloneDataStructure	2607
6.536.3.2 compareTo	2607
6.536.3.3 copyDataStructure	2607
6.536.3.4 equals	2608
6.536.3.5 equals	2608
6.536.3.6 getConnectionId	2608
6.536.3.7 getConnectionId	2608
6.536.3.8 getDataStructureType	2608
6.536.3.9 getParentId	2609
6.536.3.10getSessionId	2609
6.536.3.11getValue	2609
6.536.3.12operator<	2609
6.536.3.13operator=	2609
6.536.3.14operator==	2609
6.536.3.15setConnectionId	2609
6.536.3.16setSessionId	2609
6.536.3.17setValue	2609
6.536.3.18oString	2609
6.536.4 Field Documentation	2609
6.536.4.1 connectionId	2609

6.536.4.2 ID_PRODUCERID	2609
6.536.4.3 sessionId	2610
6.536.4.4 value	2610
6.537activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference	2611
6.537.1 Detailed Description	2611
6.537.2 Constructor & Destructor Documentation	2612
6.537.2.1 ProducerIdMarshaller	2612
6.537.2.2 ~ProducerIdMarshaller	2612
6.537.3 Member Function Documentation	2612
6.537.3.1 createObject	2612
6.537.3.2 getDataStructureType	2612
6.537.3.3 looseMarshal	2612
6.537.3.4 looseUnmarshal	2613
6.537.3.5 tightMarshal1	2613
6.537.3.6 tightMarshal2	2613
6.537.3.7 tightUnmarshal	2614
6.538activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference	2615
6.538.1 Detailed Description	2615
6.538.2 Constructor & Destructor Documentation	2616
6.538.2.1 ProducerIdMarshaller	2616
6.538.2.2 ~ProducerIdMarshaller	2616
6.538.3 Member Function Documentation	2616
6.538.3.1 createObject	2616
6.538.3.2 getDataStructureType	2616
6.538.3.3 looseMarshal	2616
6.538.3.4 looseUnmarshal	2617
6.538.3.5 tightMarshal1	2617
6.538.3.6 tightMarshal2	2617
6.538.3.7 tightUnmarshal	2618
6.539activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference	2619
6.539.1 Detailed Description	2619
6.539.2 Constructor & Destructor Documentation	2620
6.539.2.1 ProducerIdMarshaller	2620
6.539.2.2 ~ProducerIdMarshaller	2620
6.539.3 Member Function Documentation	2620
6.539.3.1 createObject	2620

6.539.3.2	getDataStructureType	2620
6.539.3.3	looseMarshal	2620
6.539.3.4	looseUnmarshal	2621
6.539.3.5	tightMarshal1	2621
6.539.3.6	tightMarshal2	2621
6.539.3.7	tightUnmarshal	2622
6.540	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference	2623
6.540.1	Detailed Description	2623
6.540.2	Constructor & Destructor Documentation	2624
6.540.2.1	ProducerIdMarshaller	2624
6.540.2.2	~ProducerIdMarshaller	2624
6.540.3	Member Function Documentation	2624
6.540.3.1	createObject	2624
6.540.3.2	getDataStructureType	2624
6.540.3.3	looseMarshal	2624
6.540.3.4	looseUnmarshal	2625
6.540.3.5	tightMarshal1	2625
6.540.3.6	tightMarshal2	2625
6.540.3.7	tightUnmarshal	2626
6.541	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference	2627
6.541.1	Detailed Description	2627
6.541.2	Constructor & Destructor Documentation	2628
6.541.2.1	ProducerIdMarshaller	2628
6.541.2.2	~ProducerIdMarshaller	2628
6.541.3	Member Function Documentation	2628
6.541.3.1	createObject	2628
6.541.3.2	getDataStructureType	2628
6.541.3.3	looseMarshal	2628
6.541.3.4	looseUnmarshal	2629
6.541.3.5	tightMarshal1	2629
6.541.3.6	tightMarshal2	2629
6.541.3.7	tightUnmarshal	2630
6.542	activemq::commands::ProducerInfo Class Reference	2631
6.542.1	Constructor & Destructor Documentation	2632
6.542.1.1	ProducerInfo	2632
6.542.1.2	ProducerInfo	2632

6.542.1.3 ~ProducerInfo	2632
6.542.2 Member Function Documentation	2632
6.542.2.1 cloneDataStructure	2632
6.542.2.2 copyDataStructure	2632
6.542.2.3 equals	2633
6.542.2.4 getBrokerPath	2633
6.542.2.5 getBrokerPath	2633
6.542.2.6 getDataStructureType	2633
6.542.2.7 getDestination	2634
6.542.2.8 getDestination	2634
6.542.2.9 getProducerId	2634
6.542.2.10 getProducerId	2634
6.542.2.11 getWindowSize	2634
6.542.2.12 sDispatchAsync	2634
6.542.2.13 sProducerInfo	2634
6.542.2.14 operator=	2634
6.542.2.15 setBrokerPath	2634
6.542.2.16 setDestination	2634
6.542.2.17 setDispatchAsync	2634
6.542.2.18 setProducerId	2634
6.542.2.19 setWindowSize	2634
6.542.2.20 toString	2634
6.542.2.21 visit	2635
6.542.3 Field Documentation	2635
6.542.3.1 brokerPath	2635
6.542.3.2 destination	2635
6.542.3.3 dispatchAsync	2635
6.542.3.4 ID_PRODUCERINFO	2635
6.542.3.5 producerId	2635
6.542.3.6 windowSize	2635
6.543 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference	2636
6.543.1 Detailed Description	2636
6.543.2 Constructor & Destructor Documentation	2637
6.543.2.1 ProducerInfoMarshaller	2637
6.543.2.2 ~ProducerInfoMarshaller	2637
6.543.3 Member Function Documentation	2637

6.543.3.1 createObject	2637
6.543.3.2 getDataStructureType	2637
6.543.3.3 looseMarshal	2637
6.543.3.4 looseUnmarshal	2638
6.543.3.5 tightMarshal1	2638
6.543.3.6 tightMarshal2	2638
6.543.3.7 tightUnmarshal	2639
6.544activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference	2640
6.544.1 Detailed Description	2640
6.544.2 Constructor & Destructor Documentation	2641
6.544.2.1 ProducerInfoMarshaller	2641
6.544.2.2 ~ProducerInfoMarshaller	2641
6.544.3 Member Function Documentation	2641
6.544.3.1 createObject	2641
6.544.3.2 getDataStructureType	2641
6.544.3.3 looseMarshal	2641
6.544.3.4 looseUnmarshal	2642
6.544.3.5 tightMarshal1	2642
6.544.3.6 tightMarshal2	2642
6.544.3.7 tightUnmarshal	2643
6.545activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference	2644
6.545.1 Detailed Description	2644
6.545.2 Constructor & Destructor Documentation	2645
6.545.2.1 ProducerInfoMarshaller	2645
6.545.2.2 ~ProducerInfoMarshaller	2645
6.545.3 Member Function Documentation	2645
6.545.3.1 createObject	2645
6.545.3.2 getDataStructureType	2645
6.545.3.3 looseMarshal	2645
6.545.3.4 looseUnmarshal	2646
6.545.3.5 tightMarshal1	2646
6.545.3.6 tightMarshal2	2646
6.545.3.7 tightUnmarshal	2647
6.546activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference	2648

6.546.1 Detailed Description	2648
6.546.2 Constructor & Destructor Documentation	2649
6.546.2.1 ProducerInfoMarshaller	2649
6.546.2.2 ~ProducerInfoMarshaller	2649
6.546.3 Member Function Documentation	2649
6.546.3.1 createObject	2649
6.546.3.2 getDataStructureType	2649
6.546.3.3 looseMarshal	2649
6.546.3.4 looseUnmarshal	2650
6.546.3.5 tightMarshal1	2650
6.546.3.6 tightMarshal2	2650
6.546.3.7 tightUnmarshal	2651
6.547activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference	2652
6.547.1 Detailed Description	2652
6.547.2 Constructor & Destructor Documentation	2653
6.547.2.1 ProducerInfoMarshaller	2653
6.547.2.2 ~ProducerInfoMarshaller	2653
6.547.3 Member Function Documentation	2653
6.547.3.1 createObject	2653
6.547.3.2 getDataStructureType	2653
6.547.3.3 looseMarshal	2653
6.547.3.4 looseUnmarshal	2654
6.547.3.5 tightMarshal1	2654
6.547.3.6 tightMarshal2	2654
6.547.3.7 tightUnmarshal	2655
6.548activemq::state::ProducerState Class Reference	2656
6.548.1 Constructor & Destructor Documentation	2656
6.548.1.1 ProducerState	2656
6.548.1.2 ~ProducerState	2656
6.548.2 Member Function Documentation	2656
6.548.2.1 getInfo	2656
6.548.2.2 toString	2656
6.549decaf::util::Properties Class Reference	2657
6.549.1 Detailed Description	2658
6.549.2 Constructor & Destructor Documentation	2659
6.549.2.1 Properties	2659

6.549.2.2 Properties	2659
6.549.2.3 ~Properties	2659
6.549.3 Member Function Documentation	2659
6.549.3.1 clear	2659
6.549.3.2 clone	2659
6.549.3.3 copy	2659
6.549.3.4 equals	2659
6.549.3.5 getProperty	2659
6.549.3.6 getProperty	2660
6.549.3.7 hasProperty	2660
6.549.3.8 isEmpty	2660
6.549.3.9 load	2660
6.549.3.10load	2662
6.549.3.11operator=	2663
6.549.3.12remove	2663
6.549.3.13setProperty	2663
6.549.3.14size	2663
6.549.3.15store	2663
6.549.3.16store	2664
6.549.3.17toArray	2665
6.549.3.18toString	2665
6.549.4 Field Documentation	2665
6.549.4.1 defaults	2665
6.550decaf::util::logging::PropertiesChangeListener Class Reference	2666
6.550.1 Detailed Description	2666
6.550.2 Constructor & Destructor Documentation	2666
6.550.2.1 ~PropertiesChangeListener	2666
6.550.3 Member Function Documentation	2666
6.550.3.1 onPropertyChanged	2666
6.551decaf::net::ProtocolException Class Reference	2667
6.551.1 Constructor & Destructor Documentation	2667
6.551.1.1 ProtocolException	2667
6.551.1.2 ProtocolException	2667
6.551.1.3 ProtocolException	2668
6.551.1.4 ProtocolException	2668
6.551.1.5 ProtocolException	2668

6.551.1.6 ProtocolException	2668
6.551.1.7 ~ProtocolException	2669
6.551.2 Member Function Documentation	2669
6.551.2.1 clone	2669
6.552decaf::security::PublicKey Class Reference	2670
6.552.1 Detailed Description	2670
6.552.2 Constructor & Destructor Documentation	2670
6.552.2.1 ~PublicKey	2670
6.553decaf::util::Queue< E > Class Template Reference	2671
6.553.1 Detailed Description	2671
6.553.2 Constructor & Destructor Documentation	2672
6.553.2.1 ~Queue	2672
6.553.3 Member Function Documentation	2672
6.553.3.1 element	2672
6.553.3.2 offer	2672
6.553.3.3 peek	2672
6.553.3.4 poll	2673
6.553.3.5 remove	2673
6.554cms::Queue Class Reference	2674
6.554.1 Detailed Description	2674
6.554.2 Constructor & Destructor Documentation	2674
6.554.2.1 ~Queue	2674
6.554.3 Member Function Documentation	2674
6.554.3.1 getQueueName	2674
6.555cms::QueueBrowser Class Reference	2675
6.555.1 Detailed Description	2675
6.555.2 Constructor & Destructor Documentation	2675
6.555.2.1 ~QueueBrowser	2675
6.555.3 Member Function Documentation	2675
6.555.3.1 getEnumeration	2675
6.555.3.2 getMessageSelector	2676
6.555.3.3 getQueue	2676
6.556decaf::util::Random Class Reference	2677
6.556.1 Detailed Description	2678
6.556.2 Constructor & Destructor Documentation	2678
6.556.2.1 Random	2678

6.556.2.2 Random	2678
6.556.3 Member Function Documentation	2678
6.556.3.1 next	2678
6.556.3.2 nextBoolean	2679
6.556.3.3 nextBytes	2679
6.556.3.4 nextDouble	2679
6.556.3.5 nextFloat	2679
6.556.3.6 nextGaussian	2680
6.556.3.7 nextInt	2680
6.556.3.8 nextInt	2680
6.556.3.9 nextLong	2680
6.556.3.10 setSeed	2681
6.557activemq::transport::inactivity::ReadChecker Class Reference	2682
6.557.1 Detailed Description	2682
6.557.2 Constructor & Destructor Documentation	2682
6.557.2.1 ReadChecker	2682
6.557.2.2 ~ReadChecker	2682
6.557.3 Member Function Documentation	2682
6.557.3.1 run	2682
6.558decaf::io::Reader Class Reference	2683
6.558.1 Constructor & Destructor Documentation	2683
6.558.1.1 ~Reader	2683
6.558.2 Member Function Documentation	2683
6.558.2.1 getInputStream	2683
6.558.2.2 read	2683
6.558.2.3 readByte	2684
6.558.2.4 setInputStream	2684
6.559decaf::nio::ReadOnlyBufferException Class Reference	2685
6.559.1 Constructor & Destructor Documentation	2685
6.559.1.1 ReadOnlyBufferException	2685
6.559.1.2 ReadOnlyBufferException	2685
6.559.1.3 ReadOnlyBufferException	2686
6.559.1.4 ReadOnlyBufferException	2686
6.559.1.5 ReadOnlyBufferException	2686
6.559.1.6 ReadOnlyBufferException	2686
6.559.1.7 ~ReadOnlyBufferException	2687

6.559.2 Member Function Documentation	2687
6.559.2.1 clone	2687
6.560decaf::util::concurrent::locks::ReadWriteLock Class Reference	2688
6.560.1 Detailed Description	2688
6.560.2 Constructor & Destructor Documentation	2689
6.560.2.1 ~ReadWriteLock	2689
6.560.3 Member Function Documentation	2689
6.560.3.1 readLock	2689
6.560.3.2 writeLock	2689
6.561activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference	2690
6.561.1 Constructor & Destructor Documentation	2690
6.561.1.1 ReceiveExecutor	2690
6.561.1.2 ~ReceiveExecutor	2690
6.561.2 Member Function Documentation	2690
6.561.2.1 doInCms	2690
6.561.2.2 getDestination	2691
6.561.2.3 getMessage	2691
6.561.3 Field Documentation	2691
6.561.3.1 destination	2691
6.561.3.2 message	2691
6.561.3.3 noLocal	2691
6.561.3.4 parent	2691
6.561.3.5 selector	2691
6.562decaf::util::concurrent::locks::ReentrantLock Class Reference	2692
6.562.1 Detailed Description	2693
6.562.2 Constructor & Destructor Documentation	2693
6.562.2.1 ReentrantLock	2693
6.562.2.2 ~ReentrantLock	2693
6.562.3 Member Function Documentation	2693
6.562.3.1 getHoldCount	2693
6.562.3.2 isFair	2694
6.562.3.3 isHeldByCurrentThread	2694
6.562.3.4 isLocked	2694
6.562.3.5 lock	2695
6.562.3.6 lockInterruptibly	2695
6.562.3.7 newCondition	2696

6.562.3.8 toString	2696
6.562.3.9 tryLock	2696
6.562.3.10 tryLock	2697
6.562.3.11 unlock	2698
6.563 decaf::util::concurrent::RejectedExecutionException Class Reference	2699
6.563.1 Constructor & Destructor Documentation	2699
6.563.1.1 RejectedExecutionException	2699
6.563.1.2 RejectedExecutionException	2699
6.563.1.3 RejectedExecutionException	2700
6.563.1.4 RejectedExecutionException	2700
6.563.1.5 RejectedExecutionException	2700
6.563.1.6 RejectedExecutionException	2700
6.563.1.7 ~RejectedExecutionException	2701
6.563.2 Member Function Documentation	2701
6.563.2.1 clone	2701
6.564 decaf::util::concurrent::RejectedExecutionHandler Class Reference	2702
6.564.1 Detailed Description	2702
6.564.2 Constructor & Destructor Documentation	2702
6.564.2.1 ~RejectedExecutionHandler	2702
6.565 activemq::commands::RemoveInfo Class Reference	2703
6.565.1 Constructor & Destructor Documentation	2704
6.565.1.1 RemoveInfo	2704
6.565.1.2 RemoveInfo	2704
6.565.1.3 ~RemoveInfo	2704
6.565.2 Member Function Documentation	2704
6.565.2.1 cloneDataStructure	2704
6.565.2.2 copyDataStructure	2704
6.565.2.3 equals	2704
6.565.2.4 getDataStructureType	2705
6.565.2.5 getLastDeliveredSequenceId	2705
6.565.2.6 getObjectId	2705
6.565.2.7 getObjectId	2705
6.565.2.8 isRemoveInfo	2705
6.565.2.9 operator=	2705
6.565.2.10 setLastDeliveredSequenceId	2705
6.565.2.11 setObjectId	2705

6.565.2.12	toString	2705
6.565.2.13	visit	2706
6.565.3	Field Documentation	2706
6.565.3.1	ID_REMOVEINFO	2706
6.565.3.2	lastDeliveredSequenceId	2706
6.565.3.3	objectId	2706
6.566	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference	2707
6.566.1	Detailed Description	2707
6.566.2	Constructor & Destructor Documentation	2708
6.566.2.1	RemoveInfoMarshaller	2708
6.566.2.2	~RemoveInfoMarshaller	2708
6.566.3	Member Function Documentation	2708
6.566.3.1	createObject	2708
6.566.3.2	getDataStructureType	2708
6.566.3.3	looseMarshal	2708
6.566.3.4	looseUnmarshal	2709
6.566.3.5	tightMarshal1	2709
6.566.3.6	tightMarshal2	2709
6.566.3.7	tightUnmarshal	2710
6.567	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference	2711
6.567.1	Detailed Description	2711
6.567.2	Constructor & Destructor Documentation	2712
6.567.2.1	RemoveInfoMarshaller	2712
6.567.2.2	~RemoveInfoMarshaller	2712
6.567.3	Member Function Documentation	2712
6.567.3.1	createObject	2712
6.567.3.2	getDataStructureType	2712
6.567.3.3	looseMarshal	2712
6.567.3.4	looseUnmarshal	2713
6.567.3.5	tightMarshal1	2713
6.567.3.6	tightMarshal2	2713
6.567.3.7	tightUnmarshal	2714
6.568	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference	2715
6.568.1	Detailed Description	2715
6.568.2	Constructor & Destructor Documentation	2716
6.568.2.1	RemoveInfoMarshaller	2716

6.568.2.2 ~RemoveInfoMarshaller	2716
6.568.3 Member Function Documentation	2716
6.568.3.1 createObject	2716
6.568.3.2 getDataStructureType	2716
6.568.3.3 looseMarshal	2716
6.568.3.4 looseUnmarshal	2717
6.568.3.5 tightMarshal1	2717
6.568.3.6 tightMarshal2	2717
6.568.3.7 tightUnmarshal	2718
6.569activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference	2719
6.569.1 Detailed Description	2719
6.569.2 Constructor & Destructor Documentation	2720
6.569.2.1 RemoveInfoMarshaller	2720
6.569.2.2 ~RemoveInfoMarshaller	2720
6.569.3 Member Function Documentation	2720
6.569.3.1 createObject	2720
6.569.3.2 getDataStructureType	2720
6.569.3.3 looseMarshal	2720
6.569.3.4 looseUnmarshal	2721
6.569.3.5 tightMarshal1	2721
6.569.3.6 tightMarshal2	2721
6.569.3.7 tightUnmarshal	2722
6.570activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference	2723
6.570.1 Detailed Description	2723
6.570.2 Constructor & Destructor Documentation	2724
6.570.2.1 RemoveInfoMarshaller	2724
6.570.2.2 ~RemoveInfoMarshaller	2724
6.570.3 Member Function Documentation	2724
6.570.3.1 createObject	2724
6.570.3.2 getDataStructureType	2724
6.570.3.3 looseMarshal	2724
6.570.3.4 looseUnmarshal	2725
6.570.3.5 tightMarshal1	2725
6.570.3.6 tightMarshal2	2725
6.570.3.7 tightUnmarshal	2726
6.571activemq::commands::RemoveSubscriptionInfo Class Reference	2727

6.571.1 Constructor & Destructor Documentation	2728
6.571.1.1 RemoveSubscriptionInfo	2728
6.571.1.2 RemoveSubscriptionInfo	2728
6.571.1.3 ~RemoveSubscriptionInfo	2728
6.571.2 Member Function Documentation	2728
6.571.2.1 cloneDataStructure	2728
6.571.2.2 copyDataStructure	2728
6.571.2.3 equals	2729
6.571.2.4 getClientId	2729
6.571.2.5 getClientId	2729
6.571.2.6 getConnectionId	2729
6.571.2.7 getConnectionId	2729
6.571.2.8 getDataStructureType	2729
6.571.2.9 getSubscriptionName	2730
6.571.2.10 getSubscriptionName	2730
6.571.2.11 isRemoveSubscriptionInfo	2730
6.571.2.12 operator=	2730
6.571.2.13 setClientId	2730
6.571.2.14 setConnectionId	2730
6.571.2.15 setSubscriptionName	2730
6.571.2.16 toString	2730
6.571.2.17 visit	2730
6.571.3 Field Documentation	2731
6.571.3.1 clientId	2731
6.571.3.2 connectionId	2731
6.571.3.3 ID_REMOVESUBSCRIPTIONINFO	2731
6.571.3.4 subscriptionName	2731
6.572 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	
Class Reference	2732
6.572.1 Detailed Description	2732
6.572.2 Constructor & Destructor Documentation	2733
6.572.2.1 RemoveSubscriptionInfoMarshaller	2733
6.572.2.2 ~RemoveSubscriptionInfoMarshaller	2733
6.572.3 Member Function Documentation	2733
6.572.3.1 createObject	2733
6.572.3.2 getDataStructureType	2733
6.572.3.3 looseMarshal	2733

6.572.3.4 looseUnmarshal	2734
6.572.3.5 tightMarshal1	2734
6.572.3.6 tightMarshal2	2734
6.572.3.7 tightUnmarshal	2735
6.573activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	
Class Reference	2736
6.573.1 Detailed Description	2736
6.573.2 Constructor & Destructor Documentation	2737
6.573.2.1 RemoveSubscriptionInfoMarshaller	2737
6.573.2.2 ~RemoveSubscriptionInfoMarshaller	2737
6.573.3 Member Function Documentation	2737
6.573.3.1 createObject	2737
6.573.3.2 getDataStructureType	2737
6.573.3.3 looseMarshal	2737
6.573.3.4 looseUnmarshal	2738
6.573.3.5 tightMarshal1	2738
6.573.3.6 tightMarshal2	2738
6.573.3.7 tightUnmarshal	2739
6.574activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	
Class Reference	2740
6.574.1 Detailed Description	2740
6.574.2 Constructor & Destructor Documentation	2741
6.574.2.1 RemoveSubscriptionInfoMarshaller	2741
6.574.2.2 ~RemoveSubscriptionInfoMarshaller	2741
6.574.3 Member Function Documentation	2741
6.574.3.1 createObject	2741
6.574.3.2 getDataStructureType	2741
6.574.3.3 looseMarshal	2741
6.574.3.4 looseUnmarshal	2742
6.574.3.5 tightMarshal1	2742
6.574.3.6 tightMarshal2	2742
6.574.3.7 tightUnmarshal	2743
6.575activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
Class Reference	2744
6.575.1 Detailed Description	2744
6.575.2 Constructor & Destructor Documentation	2745
6.575.2.1 RemoveSubscriptionInfoMarshaller	2745

6.575.2.2 ~RemoveSubscriptionInfoMarshaller	2745
6.575.3 Member Function Documentation	2745
6.575.3.1 createObject	2745
6.575.3.2 getDataStructureType	2745
6.575.3.3 looseMarshal	2745
6.575.3.4 looseUnmarshal	2746
6.575.3.5 tightMarshal1	2746
6.575.3.6 tightMarshal2	2746
6.575.3.7 tightUnmarshal	2747
6.576activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	
Class Reference	2748
6.576.1 Detailed Description	2748
6.576.2 Constructor & Destructor Documentation	2749
6.576.2.1 RemoveSubscriptionInfoMarshaller	2749
6.576.2.2 ~RemoveSubscriptionInfoMarshaller	2749
6.576.3 Member Function Documentation	2749
6.576.3.1 createObject	2749
6.576.3.2 getDataStructureType	2749
6.576.3.3 looseMarshal	2749
6.576.3.4 looseUnmarshal	2750
6.576.3.5 tightMarshal1	2750
6.576.3.6 tightMarshal2	2750
6.576.3.7 tightUnmarshal	2751
6.577activemq::commands::ReplayCommand Class Reference	2752
6.577.1 Constructor & Destructor Documentation	2753
6.577.1.1 ReplayCommand	2753
6.577.1.2 ReplayCommand	2753
6.577.1.3 ~ReplayCommand	2753
6.577.2 Member Function Documentation	2753
6.577.2.1 cloneDataStructure	2753
6.577.2.2 copyDataStructure	2753
6.577.2.3 equals	2753
6.577.2.4 getDataStructureType	2754
6.577.2.5 getFirstNakNumber	2754
6.577.2.6 getLastNakNumber	2754
6.577.2.7 operator=	2754
6.577.2.8 setFirstNakNumber	2754

6.577.2.9	setLastNakNumber	2754
6.577.2.10	toString	2754
6.577.2.11	visit	2754
6.577.3	Field Documentation	2755
6.577.3.1	firstNakNumber	2755
6.577.3.2	ID_REPLAYCOMMAND	2755
6.577.3.3	lastNakNumber	2755
6.578	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class	
	Reference	2756
6.578.1	Detailed Description	2756
6.578.2	Constructor & Destructor Documentation	2757
6.578.2.1	ReplayCommandMarshaller	2757
6.578.2.2	~ReplayCommandMarshaller	2757
6.578.3	Member Function Documentation	2757
6.578.3.1	createObject	2757
6.578.3.2	getDataStructureType	2757
6.578.3.3	looseMarshal	2757
6.578.3.4	looseUnmarshal	2758
6.578.3.5	tightMarshal1	2758
6.578.3.6	tightMarshal2	2758
6.578.3.7	tightUnmarshal	2759
6.579	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller Class	
	Reference	2760
6.579.1	Detailed Description	2760
6.579.2	Constructor & Destructor Documentation	2761
6.579.2.1	ReplayCommandMarshaller	2761
6.579.2.2	~ReplayCommandMarshaller	2761
6.579.3	Member Function Documentation	2761
6.579.3.1	createObject	2761
6.579.3.2	getDataStructureType	2761
6.579.3.3	looseMarshal	2761
6.579.3.4	looseUnmarshal	2762
6.579.3.5	tightMarshal1	2762
6.579.3.6	tightMarshal2	2762
6.579.3.7	tightUnmarshal	2763
6.580	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller Class	
	Reference	2764

6.580.1 Detailed Description	2764
6.580.2 Constructor & Destructor Documentation	2765
6.580.2.1 ReplayCommandMarshaller	2765
6.580.2.2 ~ReplayCommandMarshaller	2765
6.580.3 Member Function Documentation	2765
6.580.3.1 createObject	2765
6.580.3.2 getDataStructureType	2765
6.580.3.3 looseMarshal	2765
6.580.3.4 looseUnmarshal	2766
6.580.3.5 tightMarshal1	2766
6.580.3.6 tightMarshal2	2766
6.580.3.7 tightUnmarshal	2767
6.581activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class	
Reference	2768
6.581.1 Detailed Description	2768
6.581.2 Constructor & Destructor Documentation	2769
6.581.2.1 ReplayCommandMarshaller	2769
6.581.2.2 ~ReplayCommandMarshaller	2769
6.581.3 Member Function Documentation	2769
6.581.3.1 createObject	2769
6.581.3.2 getDataStructureType	2769
6.581.3.3 looseMarshal	2769
6.581.3.4 looseUnmarshal	2770
6.581.3.5 tightMarshal1	2770
6.581.3.6 tightMarshal2	2770
6.581.3.7 tightUnmarshal	2771
6.582activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class	
Reference	2772
6.582.1 Detailed Description	2772
6.582.2 Constructor & Destructor Documentation	2773
6.582.2.1 ReplayCommandMarshaller	2773
6.582.2.2 ~ReplayCommandMarshaller	2773
6.582.3 Member Function Documentation	2773
6.582.3.1 createObject	2773
6.582.3.2 getDataStructureType	2773
6.582.3.3 looseMarshal	2773
6.582.3.4 looseUnmarshal	2774

6.582.3.5 tightMarshal1	2774
6.582.3.6 tightMarshal2	2774
6.582.3.7 tightUnmarshal	2775
6.583activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference	2776
6.583.1 Constructor & Destructor Documentation	2776
6.583.1.1 ResolveProducerExecutor	2776
6.583.1.2 ~ResolveProducerExecutor	2776
6.583.2 Member Function Documentation	2776
6.583.2.1 getDestination	2776
6.584activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference	2777
6.584.1 Constructor & Destructor Documentation	2777
6.584.1.1 ResolveReceiveExecutor	2777
6.584.1.2 ~ResolveReceiveExecutor	2777
6.584.2 Member Function Documentation	2777
6.584.2.1 getDestination	2777
6.585activemq::cmsutil::ResourceLifecycleManager Class Reference	2778
6.585.1 Detailed Description	2778
6.585.2 Constructor & Destructor Documentation	2778
6.585.2.1 ResourceLifecycleManager	2778
6.585.2.2 ~ResourceLifecycleManager	2778
6.585.3 Member Function Documentation	2779
6.585.3.1 addConnection	2779
6.585.3.2 addDestination	2779
6.585.3.3 addMessageConsumer	2779
6.585.3.4 addMessageProducer	2779
6.585.3.5 addSession	2779
6.585.3.6 destroy	2780
6.585.3.7 releaseAll	2780
6.586activemq::commands::Response Class Reference	2781
6.586.1 Constructor & Destructor Documentation	2782
6.586.1.1 Response	2782
6.586.1.2 Response	2782
6.586.1.3 ~Response	2782
6.586.2 Member Function Documentation	2782
6.586.2.1 cloneDataStructure	2782
6.586.2.2 copyDataStructure	2782

6.586.2.3 equals	2782
6.586.2.4 getCorrelationId	2783
6.586.2.5 getDataStructureType	2783
6.586.2.6 isResponse	2783
6.586.2.7 operator=	2783
6.586.2.8 setCorrelationId	2783
6.586.2.9 toString	2783
6.586.2.10 visit	2784
6.586.3 Field Documentation	2784
6.586.3.1 correlationId	2784
6.586.3.2 ID_RESPONSE	2784
6.587activemq::transport::mock::ResponseBuilder Class Reference	2785
6.587.1 Detailed Description	2785
6.587.2 Constructor & Destructor Documentation	2785
6.587.2.1 ~ResponseBuilder	2785
6.587.3 Member Function Documentation	2785
6.587.3.1 buildIncomingCommands	2785
6.587.3.2 buildResponse	2786
6.588activemq::transport::correlator::ResponseCorrelator Class Reference	2787
6.588.1 Detailed Description	2787
6.588.2 Constructor & Destructor Documentation	2788
6.588.2.1 ResponseCorrelator	2788
6.588.2.2 ~ResponseCorrelator	2788
6.588.3 Member Function Documentation	2788
6.588.3.1 close	2788
6.588.3.2 onCommand	2788
6.588.3.3 oneway	2788
6.588.3.4 onTransportException	2789
6.588.3.5 request	2789
6.588.3.6 request	2789
6.588.3.7 start	2790
6.589activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference	2791
6.589.1 Detailed Description	2791
6.589.2 Constructor & Destructor Documentation	2792
6.589.2.1 ResponseMarshaller	2792
6.589.2.2 ~ResponseMarshaller	2792

6.589.3 Member Function Documentation	2792
6.589.3.1 createObject	2792
6.589.3.2 getDataStructureType	2792
6.589.3.3 looseMarshal	2792
6.589.3.4 looseUnmarshal	2793
6.589.3.5 tightMarshal1	2793
6.589.3.6 tightMarshal2	2794
6.589.3.7 tightUnmarshal	2795
6.590activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference .	2796
6.590.1 Detailed Description	2796
6.590.2 Constructor & Destructor Documentation	2797
6.590.2.1 ResponseMarshaller	2797
6.590.2.2 ~ResponseMarshaller	2797
6.590.3 Member Function Documentation	2797
6.590.3.1 createObject	2797
6.590.3.2 getDataStructureType	2797
6.590.3.3 looseMarshal	2797
6.590.3.4 looseUnmarshal	2798
6.590.3.5 tightMarshal1	2798
6.590.3.6 tightMarshal2	2799
6.590.3.7 tightUnmarshal	2800
6.591activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference .	2801
6.591.1 Detailed Description	2801
6.591.2 Constructor & Destructor Documentation	2802
6.591.2.1 ResponseMarshaller	2802
6.591.2.2 ~ResponseMarshaller	2802
6.591.3 Member Function Documentation	2802
6.591.3.1 createObject	2802
6.591.3.2 getDataStructureType	2802
6.591.3.3 looseMarshal	2802
6.591.3.4 looseUnmarshal	2803
6.591.3.5 tightMarshal1	2803
6.591.3.6 tightMarshal2	2804
6.591.3.7 tightUnmarshal	2805
6.592activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference .	2806
6.592.1 Detailed Description	2806

6.592.2 Constructor & Destructor Documentation	2807
6.592.2.1 ResponseMarshaller	2807
6.592.2.2 ~ResponseMarshaller	2807
6.592.3 Member Function Documentation	2807
6.592.3.1 createObject	2807
6.592.3.2 getDataStructureType	2807
6.592.3.3 looseMarshal	2807
6.592.3.4 looseUnmarshal	2808
6.592.3.5 tightMarshal1	2808
6.592.3.6 tightMarshal2	2809
6.592.3.7 tightUnmarshal	2810
6.593activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference	2811
6.593.1 Detailed Description	2811
6.593.2 Constructor & Destructor Documentation	2812
6.593.2.1 ResponseMarshaller	2812
6.593.2.2 ~ResponseMarshaller	2812
6.593.3 Member Function Documentation	2812
6.593.3.1 createObject	2812
6.593.3.2 getDataStructureType	2812
6.593.3.3 looseMarshal	2812
6.593.3.4 looseUnmarshal	2813
6.593.3.5 tightMarshal1	2813
6.593.3.6 tightMarshal2	2814
6.593.3.7 tightUnmarshal	2815
6.594decaf::lang::Runnable Class Reference	2816
6.594.1 Detailed Description	2816
6.594.2 Constructor & Destructor Documentation	2816
6.594.2.1 ~Runnable	2816
6.594.3 Member Function Documentation	2816
6.594.3.1 run	2816
6.595decaf::lang::Runtime Class Reference	2817
6.595.1 Constructor & Destructor Documentation	2817
6.595.1.1 ~Runtime	2817
6.595.2 Member Function Documentation	2817
6.595.2.1 getRuntime	2817
6.595.2.2 initializeRuntime	2817

6.595.2.3 initializeRuntime	2818
6.595.2.4 shutdownRuntime	2818
6.596decaf::lang::exceptions::RuntimeException Class Reference	2819
6.596.1 Constructor & Destructor Documentation	2819
6.596.1.1 RuntimeException	2819
6.596.1.2 RuntimeException	2819
6.596.1.3 RuntimeException	2820
6.596.1.4 RuntimeException	2820
6.596.1.5 RuntimeException	2820
6.596.1.6 RuntimeException	2820
6.596.1.7 ~RuntimeException	2821
6.596.2 Member Function Documentation	2821
6.596.2.1 clone	2821
6.597decaf::security_provider::SecurityProvider Class Reference	2822
6.597.1 Constructor & Destructor Documentation	2822
6.597.1.1 ~SecurityProvider	2822
6.597.2 Member Function Documentation	2822
6.597.2.1 createX500Principal	2822
6.597.2.2 createX500Principal	2822
6.597.2.3 createX500Principal	2822
6.598decaf::security_provider::SecurityProviderMap Class Reference	2823
6.598.1 Detailed Description	2823
6.598.2 Member Function Documentation	2823
6.598.2.1 getInstance	2823
6.598.2.2 getSecurityProviderNames	2823
6.598.2.3 lookup	2824
6.598.2.4 registerSecurityProvider	2824
6.598.2.5 unregisterSecurityProvider	2824
6.599decaf::security_provider::SecurityProviderRegistrar Class Reference	2825
6.599.1 Detailed Description	2825
6.599.2 Constructor & Destructor Documentation	2825
6.599.2.1 SecurityProviderRegistrar	2825
6.599.2.2 ~SecurityProviderRegistrar	2825
6.599.3 Member Function Documentation	2826
6.599.3.1 getProvider	2826
6.600decaf::util::concurrent::Semaphore Class Reference	2827

6.600.1 Detailed Description	2828
6.600.2 Constructor & Destructor Documentation	2829
6.600.2.1 Semaphore	2829
6.600.2.2 Semaphore	2829
6.600.2.3 ~Semaphore	2830
6.600.3 Member Function Documentation	2830
6.600.3.1 acquire	2830
6.600.3.2 acquire	2830
6.600.3.3 acquireUninterruptibly	2831
6.600.3.4 acquireUninterruptibly	2831
6.600.3.5 availablePermits	2832
6.600.3.6 drainPermits	2832
6.600.3.7 isFair	2832
6.600.3.8 release	2832
6.600.3.9 release	2833
6.600.3.10oString	2833
6.600.3.11tryAcquire	2833
6.600.3.12ryAcquire	2834
6.600.3.13ryAcquire	2834
6.600.3.14ryAcquire	2835
6.601activemq::cmsutil::CmsTemplate::SendExecutor Class Reference	2836
6.601.1 Constructor & Destructor Documentation	2836
6.601.1.1 SendExecutor	2836
6.601.1.2 ~SendExecutor	2836
6.601.2 Member Function Documentation	2836
6.601.2.1 doInCms	2836
6.602decaf::net::ServerSocket Class Reference	2837
6.602.1 Detailed Description	2837
6.602.2 Member Typedef Documentation	2837
6.602.2.1 SocketAddress	2837
6.602.2.2 SocketHandle	2837
6.602.3 Constructor & Destructor Documentation	2837
6.602.3.1 ServerSocket	2837
6.602.3.2 ~ServerSocket	2838
6.602.4 Member Function Documentation	2838
6.602.4.1 accept	2838

6.602.4.2 bind	2838
6.602.4.3 bind	2838
6.602.4.4 close	2838
6.602.4.5 isBound	2838
6.603cms::Session Class Reference	2839
6.603.1 Detailed Description	2841
6.603.2 Member Enumeration Documentation	2842
6.603.2.1 AcknowledgeMode	2842
6.603.3 Constructor & Destructor Documentation	2842
6.603.3.1 ~Session	2842
6.603.4 Member Function Documentation	2842
6.603.4.1 close	2842
6.603.4.2 commit	2843
6.603.4.3 createBrowser	2843
6.603.4.4 createBrowser	2843
6.603.4.5 createBytesMessage	2844
6.603.4.6 createBytesMessage	2844
6.603.4.7 createConsumer	2844
6.603.4.8 createConsumer	2845
6.603.4.9 createConsumer	2845
6.603.4.10createDurableConsumer	2846
6.603.4.11createMapMessage	2846
6.603.4.12reateMessage	2847
6.603.4.13reateProducer	2847
6.603.4.14reateQueue	2847
6.603.4.15createStreamMessage	2848
6.603.4.16reateTemporaryQueue	2848
6.603.4.17reateTemporaryTopic	2848
6.603.4.18reateTextMessage	2848
6.603.4.19reateTextMessage	2849
6.603.4.20reateTopic	2849
6.603.4.21getAcknowledgeMode	2849
6.603.4.22sTransacted	2850
6.603.4.23recover	2850
6.603.4.24rollback	2850
6.603.4.25unsubscribe	2851

6.604	activemq::cmsutil::SessionCallback Class Reference	2852
6.604.1	Detailed Description	2852
6.604.2	Constructor & Destructor Documentation	2852
6.604.2.1	~SessionCallback	2852
6.604.3	Member Function Documentation	2852
6.604.3.1	doInCms	2852
6.605	activemq::commands::SessionId Class Reference	2853
6.605.1	Member Typedef Documentation	2854
6.605.1.1	COMPARATOR	2854
6.605.2	Constructor & Destructor Documentation	2854
6.605.2.1	SessionId	2854
6.605.2.2	SessionId	2854
6.605.2.3	SessionId	2854
6.605.2.4	SessionId	2854
6.605.2.5	SessionId	2854
6.605.2.6	~SessionId	2854
6.605.3	Member Function Documentation	2854
6.605.3.1	cloneDataStructure	2854
6.605.3.2	compareTo	2854
6.605.3.3	copyDataStructure	2854
6.605.3.4	equals	2855
6.605.3.5	equals	2855
6.605.3.6	getConnectionId	2855
6.605.3.7	getConnectionId	2855
6.605.3.8	getDataStructureType	2855
6.605.3.9	getParentId	2855
6.605.3.10	getValue	2855
6.605.3.11	operator<	2856
6.605.3.12	operator=	2856
6.605.3.13	operator==	2856
6.605.3.14	setConnectionId	2856
6.605.3.15	setValue	2856
6.605.3.16	toString	2856
6.605.4	Field Documentation	2856
6.605.4.1	connectionId	2856
6.605.4.2	ID_SESSIONID	2856

6.605.4.3 value	2856
6.606activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference	2857
6.606.1 Detailed Description	2857
6.606.2 Constructor & Destructor Documentation	2858
6.606.2.1 SessionIdMarshaller	2858
6.606.2.2 ~SessionIdMarshaller	2858
6.606.3 Member Function Documentation	2858
6.606.3.1 createObject	2858
6.606.3.2 getDataStructureType	2858
6.606.3.3 looseMarshal	2858
6.606.3.4 looseUnmarshal	2859
6.606.3.5 tightMarshal1	2859
6.606.3.6 tightMarshal2	2859
6.606.3.7 tightUnmarshal	2860
6.607activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference	2861
6.607.1 Detailed Description	2861
6.607.2 Constructor & Destructor Documentation	2862
6.607.2.1 SessionIdMarshaller	2862
6.607.2.2 ~SessionIdMarshaller	2862
6.607.3 Member Function Documentation	2862
6.607.3.1 createObject	2862
6.607.3.2 getDataStructureType	2862
6.607.3.3 looseMarshal	2862
6.607.3.4 looseUnmarshal	2863
6.607.3.5 tightMarshal1	2863
6.607.3.6 tightMarshal2	2863
6.607.3.7 tightUnmarshal	2864
6.608activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference	2865
6.608.1 Detailed Description	2865
6.608.2 Constructor & Destructor Documentation	2866
6.608.2.1 SessionIdMarshaller	2866
6.608.2.2 ~SessionIdMarshaller	2866
6.608.3 Member Function Documentation	2866
6.608.3.1 createObject	2866
6.608.3.2 getDataStructureType	2866
6.608.3.3 looseMarshal	2866

6.608.3.4 looseUnmarshal	2867
6.608.3.5 tightMarshal1	2867
6.608.3.6 tightMarshal2	2867
6.608.3.7 tightUnmarshal	2868
6.609activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference .	2869
6.609.1 Detailed Description	2869
6.609.2 Constructor & Destructor Documentation	2870
6.609.2.1 SessionIdMarshaller	2870
6.609.2.2 ~SessionIdMarshaller	2870
6.609.3 Member Function Documentation	2870
6.609.3.1 createObject	2870
6.609.3.2 getDataStructureType	2870
6.609.3.3 looseMarshal	2870
6.609.3.4 looseUnmarshal	2871
6.609.3.5 tightMarshal1	2871
6.609.3.6 tightMarshal2	2871
6.609.3.7 tightUnmarshal	2872
6.610activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference .	2873
6.610.1 Detailed Description	2873
6.610.2 Constructor & Destructor Documentation	2874
6.610.2.1 SessionIdMarshaller	2874
6.610.2.2 ~SessionIdMarshaller	2874
6.610.3 Member Function Documentation	2874
6.610.3.1 createObject	2874
6.610.3.2 getDataStructureType	2874
6.610.3.3 looseMarshal	2874
6.610.3.4 looseUnmarshal	2875
6.610.3.5 tightMarshal1	2875
6.610.3.6 tightMarshal2	2875
6.610.3.7 tightUnmarshal	2876
6.611activemq::commands::SessionInfo Class Reference	2877
6.611.1 Constructor & Destructor Documentation	2878
6.611.1.1 SessionInfo	2878
6.611.1.2 SessionInfo	2878
6.611.1.3 ~SessionInfo	2878
6.611.2 Member Function Documentation	2878

6.611.2.1 cloneDataStructure	2878
6.611.2.2 copyDataStructure	2878
6.611.2.3 equals	2878
6.611.2.4 getAckMode	2879
6.611.2.5 getDataStructureType	2879
6.611.2.6 getSessionId	2879
6.611.2.7 getSessionId	2879
6.611.2.8 operator=	2879
6.611.2.9 setAckMode	2879
6.611.2.10 setSessionId	2879
6.611.2.11 toString	2879
6.611.2.12 visit	2879
6.611.3 Field Documentation	2880
6.611.3.1 ID_SESSIONINFO	2880
6.611.3.2 sessionId	2880
6.612activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference	2881
6.612.1 Detailed Description	2881
6.612.2 Constructor & Destructor Documentation	2882
6.612.2.1 SessionInfoMarshaller	2882
6.612.2.2 ~SessionInfoMarshaller	2882
6.612.3 Member Function Documentation	2882
6.612.3.1 createObject	2882
6.612.3.2 getDataStructureType	2882
6.612.3.3 looseMarshal	2882
6.612.3.4 looseUnmarshal	2883
6.612.3.5 tightMarshal1	2883
6.612.3.6 tightMarshal2	2883
6.612.3.7 tightUnmarshal	2884
6.613activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference	2885
6.613.1 Detailed Description	2885
6.613.2 Constructor & Destructor Documentation	2886
6.613.2.1 SessionInfoMarshaller	2886
6.613.2.2 ~SessionInfoMarshaller	2886
6.613.3 Member Function Documentation	2886
6.613.3.1 createObject	2886
6.613.3.2 getDataStructureType	2886

6.613.3.3 looseMarshal	2886
6.613.3.4 looseUnmarshal	2887
6.613.3.5 tightMarshal1	2887
6.613.3.6 tightMarshal2	2887
6.613.3.7 tightUnmarshal	2888
6.614activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference	2889
6.614.1 Detailed Description	2889
6.614.2 Constructor & Destructor Documentation	2890
6.614.2.1 SessionInfoMarshaller	2890
6.614.2.2 ~SessionInfoMarshaller	2890
6.614.3 Member Function Documentation	2890
6.614.3.1 createObject	2890
6.614.3.2 getDataStructureType	2890
6.614.3.3 looseMarshal	2890
6.614.3.4 looseUnmarshal	2891
6.614.3.5 tightMarshal1	2891
6.614.3.6 tightMarshal2	2891
6.614.3.7 tightUnmarshal	2892
6.615activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference	2893
6.615.1 Detailed Description	2893
6.615.2 Constructor & Destructor Documentation	2894
6.615.2.1 SessionInfoMarshaller	2894
6.615.2.2 ~SessionInfoMarshaller	2894
6.615.3 Member Function Documentation	2894
6.615.3.1 createObject	2894
6.615.3.2 getDataStructureType	2894
6.615.3.3 looseMarshal	2894
6.615.3.4 looseUnmarshal	2895
6.615.3.5 tightMarshal1	2895
6.615.3.6 tightMarshal2	2895
6.615.3.7 tightUnmarshal	2896
6.616activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference	2897
6.616.1 Detailed Description	2897
6.616.2 Constructor & Destructor Documentation	2898
6.616.2.1 SessionInfoMarshaller	2898
6.616.2.2 ~SessionInfoMarshaller	2898

6.616.3 Member Function Documentation	2898
6.616.3.1 createObject	2898
6.616.3.2 getDataStructureType	2898
6.616.3.3 looseMarshal	2898
6.616.3.4 looseUnmarshal	2899
6.616.3.5 tightMarshal1	2899
6.616.3.6 tightMarshal2	2899
6.616.3.7 tightUnmarshal	2900
6.617activemq::cmsutil::SessionPool Class Reference	2901
6.617.1 Detailed Description	2901
6.617.2 Constructor & Destructor Documentation	2901
6.617.2.1 SessionPool	2901
6.617.2.2 ~SessionPool	2901
6.617.3 Member Function Documentation	2902
6.617.3.1 getResourceLifecycleManager	2902
6.617.3.2 returnSession	2902
6.617.3.3 takeSession	2902
6.618activemq::state::SessionState Class Reference	2903
6.618.1 Constructor & Destructor Documentation	2904
6.618.1.1 SessionState	2904
6.618.1.2 ~SessionState	2904
6.618.2 Member Function Documentation	2904
6.618.2.1 addConsumer	2904
6.618.2.2 addProducer	2904
6.618.2.3 checkShutdown	2904
6.618.2.4 getConsumerState	2904
6.618.2.5 getConsumerStates	2904
6.618.2.6 getInfo	2904
6.618.2.7 getProducerState	2904
6.618.2.8 getProducerStates	2904
6.618.2.9 removeConsumer	2904
6.618.2.10removeProducer	2904
6.618.2.11shutdown	2904
6.618.2.12toString	2904
6.619decaf::util::Set< E > Class Template Reference	2905
6.619.1 Detailed Description	2905

6.619.2 Constructor & Destructor Documentation	2905
6.619.2.1 ~Set	2905
6.620decaf::lang::Short Class Reference	2906
6.620.1 Constructor & Destructor Documentation	2907
6.620.1.1 Short	2907
6.620.1.2 Short	2908
6.620.1.3 ~Short	2908
6.620.2 Member Function Documentation	2908
6.620.2.1 byteValue	2908
6.620.2.2 compareTo	2908
6.620.2.3 compareTo	2908
6.620.2.4 decode	2909
6.620.2.5 doubleValue	2909
6.620.2.6 equals	2909
6.620.2.7 equals	2909
6.620.2.8 floatValue	2910
6.620.2.9 intValue	2910
6.620.2.10longValue	2910
6.620.2.11operator<	2910
6.620.2.12operator<	2910
6.620.2.13operator==	2911
6.620.2.14operator==	2911
6.620.2.15parseShort	2911
6.620.2.16parseShort	2912
6.620.2.17reverseBytes	2912
6.620.2.18shortValue	2912
6.620.2.19oString	2912
6.620.2.20oString	2913
6.620.2.21valueOf	2913
6.620.2.22valueOf	2913
6.620.2.23valueOf	2913
6.620.3 Field Documentation	2914
6.620.3.1 MAX_VALUE	2914
6.620.3.2 MIN_VALUE	2914
6.620.3.3 SIZE	2914
6.621decaf::internal::nio::ShortArrayBuffer Class Reference	2915

6.621.1 Constructor & Destructor Documentation	2916
6.621.1.1 ShortArrayBuffer	2916
6.621.1.2 ShortArrayBuffer	2916
6.621.1.3 ShortArrayBuffer	2917
6.621.1.4 ShortArrayBuffer	2917
6.621.1.5 ~ShortArrayBuffer	2917
6.621.2 Member Function Documentation	2917
6.621.2.1 array	2917
6.621.2.2 arrayOffset	2918
6.621.2.3 asReadOnlyBuffer	2918
6.621.2.4 compact	2918
6.621.2.5 duplicate	2919
6.621.2.6 get	2919
6.621.2.7 get	2920
6.621.2.8 hasArray	2920
6.621.2.9 isReadOnly	2920
6.621.2.10put	2920
6.621.2.11put	2921
6.621.2.12setReadOnly	2921
6.621.2.13slice	2921
6.622decaf::nio::ShortBuffer Class Reference	2923
6.622.1 Detailed Description	2925
6.622.2 Constructor & Destructor Documentation	2925
6.622.2.1 ShortBuffer	2925
6.622.2.2 ~ShortBuffer	2925
6.622.3 Member Function Documentation	2925
6.622.3.1 allocate	2925
6.622.3.2 array	2926
6.622.3.3 arrayOffset	2926
6.622.3.4 asReadOnlyBuffer	2926
6.622.3.5 compact	2927
6.622.3.6 compareTo	2927
6.622.3.7 duplicate	2927
6.622.3.8 equals	2928
6.622.3.9 get	2928
6.622.3.10get	2928

6.622.3.11	get	2929
6.622.3.12	get	2929
6.622.3.13	hasArray	2929
6.622.3.14	operator<	2929
6.622.3.15	operator==	2930
6.622.3.16	put	2930
6.622.3.17	put	2930
6.622.3.18	put	2931
6.622.3.19	put	2931
6.622.3.20	put	2932
6.622.3.21	slice	2932
6.622.3.22	toString	2932
6.622.3.23	wrap	2933
6.622.3.24	wrap	2933
6.623	activemq::commands::ShutdownInfo Class Reference	2934
6.623.1	Constructor & Destructor Documentation	2935
6.623.1.1	ShutdownInfo	2935
6.623.1.2	ShutdownInfo	2935
6.623.1.3	~ShutdownInfo	2935
6.623.2	Member Function Documentation	2935
6.623.2.1	cloneDataStructure	2935
6.623.2.2	copyDataStructure	2935
6.623.2.3	equals	2935
6.623.2.4	getDataStructureType	2935
6.623.2.5	isShutdownInfo	2936
6.623.2.6	operator=	2936
6.623.2.7	toString	2936
6.623.2.8	visit	2936
6.623.3	Field Documentation	2936
6.623.3.1	ID_SHUTDOWNINFO	2936
6.624	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Refer- ence	2937
6.624.1	Detailed Description	2937
6.624.2	Constructor & Destructor Documentation	2938
6.624.2.1	ShutdownInfoMarshaller	2938
6.624.2.2	~ShutdownInfoMarshaller	2938
6.624.3	Member Function Documentation	2938

6.624.3.1	createObject	2938
6.624.3.2	getDataStructureType	2938
6.624.3.3	looseMarshal	2938
6.624.3.4	looseUnmarshal	2939
6.624.3.5	tightMarshal1	2939
6.624.3.6	tightMarshal2	2939
6.624.3.7	tightUnmarshal	2940
6.625	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference	2941
6.625.1	Detailed Description	2941
6.625.2	Constructor & Destructor Documentation	2942
6.625.2.1	ShutdownInfoMarshaller	2942
6.625.2.2	~ShutdownInfoMarshaller	2942
6.625.3	Member Function Documentation	2942
6.625.3.1	createObject	2942
6.625.3.2	getDataStructureType	2942
6.625.3.3	looseMarshal	2942
6.625.3.4	looseUnmarshal	2943
6.625.3.5	tightMarshal1	2943
6.625.3.6	tightMarshal2	2943
6.625.3.7	tightUnmarshal	2944
6.626	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference	2945
6.626.1	Detailed Description	2945
6.626.2	Constructor & Destructor Documentation	2946
6.626.2.1	ShutdownInfoMarshaller	2946
6.626.2.2	~ShutdownInfoMarshaller	2946
6.626.3	Member Function Documentation	2946
6.626.3.1	createObject	2946
6.626.3.2	getDataStructureType	2946
6.626.3.3	looseMarshal	2946
6.626.3.4	looseUnmarshal	2947
6.626.3.5	tightMarshal1	2947
6.626.3.6	tightMarshal2	2947
6.626.3.7	tightUnmarshal	2948
6.627	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Reference	2949

6.627.1 Detailed Description	2949
6.627.2 Constructor & Destructor Documentation	2950
6.627.2.1 ShutdownInfoMarshaller	2950
6.627.2.2 ~ShutdownInfoMarshaller	2950
6.627.3 Member Function Documentation	2950
6.627.3.1 createObject	2950
6.627.3.2 getDataStructureType	2950
6.627.3.3 looseMarshal	2950
6.627.3.4 looseUnmarshal	2951
6.627.3.5 tightMarshal1	2951
6.627.3.6 tightMarshal2	2951
6.627.3.7 tightUnmarshal	2952
6.628activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference	2953
6.628.1 Detailed Description	2953
6.628.2 Constructor & Destructor Documentation	2954
6.628.2.1 ShutdownInfoMarshaller	2954
6.628.2.2 ~ShutdownInfoMarshaller	2954
6.628.3 Member Function Documentation	2954
6.628.3.1 createObject	2954
6.628.3.2 getDataStructureType	2954
6.628.3.3 looseMarshal	2954
6.628.3.4 looseUnmarshal	2955
6.628.3.5 tightMarshal1	2955
6.628.3.6 tightMarshal2	2955
6.628.3.7 tightUnmarshal	2956
6.629decaf::security::SignatureException Class Reference	2957
6.629.1 Constructor & Destructor Documentation	2957
6.629.1.1 SignatureException	2957
6.629.1.2 SignatureException	2957
6.629.1.3 SignatureException	2958
6.629.1.4 SignatureException	2958
6.629.1.5 SignatureException	2958
6.629.1.6 SignatureException	2958
6.629.1.7 ~SignatureException	2959
6.629.2 Member Function Documentation	2959
6.629.2.1 clone	2959

6.630decaf::util::logging::SimpleFormatter Class Reference	2960
6.630.1 Detailed Description	2960
6.630.2 Constructor & Destructor Documentation	2960
6.630.2.1 SimpleFormatter	2960
6.630.2.2 ~SimpleFormatter	2960
6.630.3 Member Function Documentation	2960
6.630.3.1 format	2960
6.630.3.2 formatMessage	2961
6.630.3.3 getHead	2961
6.630.3.4 getTail	2961
6.631decaf::util::logging::SimpleLogger Class Reference	2962
6.631.1 Constructor & Destructor Documentation	2962
6.631.1.1 SimpleLogger	2962
6.631.1.2 ~SimpleLogger	2962
6.631.2 Member Function Documentation	2963
6.631.2.1 debug	2963
6.631.2.2 error	2963
6.631.2.3 fatal	2963
6.631.2.4 info	2963
6.631.2.5 log	2963
6.631.2.6 mark	2963
6.631.2.7 warn	2963
6.632decaf::net::Socket Class Reference	2964
6.632.1 Member Typedef Documentation	2965
6.632.1.1 SocketAddress	2965
6.632.1.2 SocketHandle	2965
6.632.2 Constructor & Destructor Documentation	2965
6.632.2.1 ~Socket	2965
6.632.3 Member Function Documentation	2965
6.632.3.1 connect	2965
6.632.3.2 getInputStream	2966
6.632.3.3 getKeepAlive	2966
6.632.3.4 getOutputStream	2966
6.632.3.5 getReceiveBufferSize	2966
6.632.3.6 getReuseAddress	2967
6.632.3.7 getSendBufferSize	2967

6.632.3.8	getSoLinger	2967
6.632.3.9	getSoTimeout	2967
6.632.3.10	isConnected	2968
6.632.3.11	setKeepAlive	2968
6.632.3.12	setReceiveBufferSize	2968
6.632.3.13	setReuseAddress	2969
6.632.3.14	setSendBufferSize	2969
6.632.3.15	setSoLinger	2969
6.632.3.16	setSoTimeout	2969
6.633	decaf::net::SocketError Class Reference	2971
6.633.1	Detailed Description	2971
6.633.2	Member Function Documentation	2971
6.633.2.1	getErrorCode	2971
6.633.2.2	getErrorString	2971
6.634	decaf::net::SocketException Class Reference	2972
6.634.1	Detailed Description	2972
6.634.2	Constructor & Destructor Documentation	2972
6.634.2.1	SocketException	2972
6.634.2.2	SocketException	2972
6.634.2.3	SocketException	2972
6.634.2.4	SocketException	2972
6.634.2.5	SocketException	2973
6.634.2.6	SocketException	2973
6.634.2.7	~SocketException	2973
6.634.3	Member Function Documentation	2973
6.634.3.1	clone	2973
6.635	decaf::net::SocketFactory Class Reference	2975
6.635.1	Detailed Description	2975
6.635.2	Constructor & Destructor Documentation	2975
6.635.2.1	~SocketFactory	2975
6.635.3	Member Function Documentation	2975
6.635.3.1	createSocket	2975
6.636	decaf::net::SocketInputStream Class Reference	2977
6.636.1	Detailed Description	2978
6.636.2	Constructor & Destructor Documentation	2978
6.636.2.1	SocketInputStream	2978

6.636.2.2 ~SocketInputStream	2978
6.636.3 Member Function Documentation	2979
6.636.3.1 available	2979
6.636.3.2 close	2979
6.636.3.3 lock	2979
6.636.3.4 mark	2979
6.636.3.5 markSupported	2979
6.636.3.6 notify	2980
6.636.3.7 notifyAll	2980
6.636.3.8 read	2980
6.636.3.9 read	2981
6.636.3.10 reset	2981
6.636.3.11 skip	2981
6.636.3.12 tryLock	2982
6.636.3.13 unlock	2982
6.636.3.14 wait	2982
6.636.3.15 wait	2983
6.636.3.16 wait	2983
6.637 decaf::net::SocketOutputStream Class Reference	2984
6.637.1 Detailed Description	2985
6.637.2 Constructor & Destructor Documentation	2985
6.637.2.1 SocketOutputStream	2985
6.637.2.2 ~SocketOutputStream	2985
6.637.3 Member Function Documentation	2985
6.637.3.1 close	2985
6.637.3.2 flush	2985
6.637.3.3 lock	2986
6.637.3.4 notify	2986
6.637.3.5 notifyAll	2986
6.637.3.6 tryLock	2987
6.637.3.7 unlock	2987
6.637.3.8 wait	2987
6.637.3.9 wait	2988
6.637.3.10 wait	2988
6.637.3.11 write	2988
6.637.3.12 write	2989

6.637.3.13	write	2989
6.638	decaf::net::SocketTimeoutException Class Reference	2990
6.638.1	Constructor & Destructor Documentation	2990
6.638.1.1	SocketTimeoutException	2990
6.638.1.2	SocketTimeoutException	2990
6.638.1.3	SocketTimeoutException	2991
6.638.1.4	SocketTimeoutException	2991
6.638.1.5	SocketTimeoutException	2991
6.638.1.6	SocketTimeoutException	2991
6.638.1.7	~SocketTimeoutException	2992
6.638.2	Member Function Documentation	2992
6.638.2.1	clone	2992
6.639	activemq::commands::BrokerError::StackTraceElement Struct Reference	2993
6.639.1	Field Documentation	2993
6.639.1.1	ClassName	2993
6.639.1.2	FileName	2993
6.639.1.3	LineNumber	2993
6.639.1.4	MethodName	2993
6.640	decaf::internal::io::StandardErrorOutputStream Class Reference	2994
6.640.1	Detailed Description	2995
6.640.2	Constructor & Destructor Documentation	2995
6.640.2.1	StandardErrorOutputStream	2995
6.640.2.2	~StandardErrorOutputStream	2995
6.640.3	Member Function Documentation	2995
6.640.3.1	close	2995
6.640.3.2	flush	2995
6.640.3.3	lock	2996
6.640.3.4	notify	2996
6.640.3.5	notifyAll	2996
6.640.3.6	tryLock	2996
6.640.3.7	unlock	2997
6.640.3.8	wait	2997
6.640.3.9	wait	2998
6.640.3.10	wait	2998
6.640.3.11	write	2998
6.640.3.12	write	2999

6.640.3.13	write	2999
6.641	decaf::internal::io::StandardInputStream Class Reference	3000
6.641.1	Constructor & Destructor Documentation	3001
6.641.1.1	StandardInputStream	3001
6.641.1.2	~StandardInputStream	3001
6.641.2	Member Function Documentation	3001
6.641.2.1	available	3001
6.641.2.2	close	3002
6.641.2.3	lock	3002
6.641.2.4	mark	3002
6.641.2.5	markSupported	3002
6.641.2.6	notify	3003
6.641.2.7	notifyAll	3003
6.641.2.8	read	3003
6.641.2.9	read	3004
6.641.2.10	reset	3004
6.641.2.11	skip	3004
6.641.2.12	tryLock	3005
6.641.2.13	unlock	3005
6.641.2.14	wait	3005
6.641.2.15	wait	3006
6.641.2.16	wait	3006
6.642	decaf::internal::io::StandardOutputStream Class Reference	3008
6.642.1	Constructor & Destructor Documentation	3009
6.642.1.1	StandardOutputStream	3009
6.642.1.2	~StandardOutputStream	3009
6.642.2	Member Function Documentation	3009
6.642.2.1	close	3009
6.642.2.2	flush	3009
6.642.2.3	lock	3009
6.642.2.4	notify	3010
6.642.2.5	notifyAll	3010
6.642.2.6	tryLock	3010
6.642.2.7	unlock	3010
6.642.2.8	wait	3011
6.642.2.9	wait	3011

6.642.2.10wait	3012
6.642.2.11write	3012
6.642.2.12write	3012
6.642.2.13write	3013
6.643cms::Startable Class Reference	3014
6.643.1 Detailed Description	3014
6.643.2 Constructor & Destructor Documentation	3014
6.643.2.1 ~Startable	3014
6.643.3 Member Function Documentation	3014
6.643.3.1 start	3014
6.644decaf::lang::STATIC_CAST_TOKEN Struct Reference	3015
6.645activemq::core::ActiveMQConstants::StaticInitializer Class Reference	3016
6.645.1 Constructor & Destructor Documentation	3016
6.645.1.1 StaticInitializer	3016
6.645.1.2 ~StaticInitializer	3016
6.645.2 Field Documentation	3016
6.645.2.1 destOptionMap	3016
6.645.2.2 destOptions	3016
6.645.2.3 uriParams	3016
6.645.2.4 uriParamsMap	3016
6.646decaf::util::StlList< E > Class Template Reference	3017
6.646.1 Detailed Description	3021
6.646.2 Constructor & Destructor Documentation	3021
6.646.2.1 StlList	3021
6.646.2.2 StlList	3021
6.646.2.3 StlList	3022
6.646.2.4 ~StlList	3022
6.646.3 Member Function Documentation	3022
6.646.3.1 add	3022
6.646.3.2 add	3022
6.646.3.3 addAll	3023
6.646.3.4 clear	3024
6.646.3.5 contains	3024
6.646.3.6 copy	3024
6.646.3.7 equals	3024
6.646.3.8 get	3024

6.646.3.9	indexOf	3025
6.646.3.10	isEmpty	3025
6.646.3.11	iterator	3025
6.646.3.12	iterator	3026
6.646.3.13	lastIndexOf	3026
6.646.3.14	listIterator	3026
6.646.3.15	listIterator	3026
6.646.3.16	listIterator	3027
6.646.3.17	listIterator	3027
6.646.3.18	remove	3027
6.646.3.19	remove	3027
6.646.3.20	set	3028
6.646.3.21	size	3028
6.647	decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference	3030
6.647.1	Detailed Description	3033
6.647.2	Constructor & Destructor Documentation	3033
6.647.2.1	StlMap	3033
6.647.2.2	StlMap	3034
6.647.2.3	StlMap	3034
6.647.2.4	~StlMap	3034
6.647.3	Member Function Documentation	3034
6.647.3.1	clear	3034
6.647.3.2	containsKey	3034
6.647.3.3	containsValue	3035
6.647.3.4	copy	3035
6.647.3.5	copy	3035
6.647.3.6	equals	3035
6.647.3.7	equals	3036
6.647.3.8	get	3036
6.647.3.9	get	3036
6.647.3.10	isEmpty	3037
6.647.3.11	keySet	3037
6.647.3.12	lock	3037
6.647.3.13	notify	3038
6.647.3.14	notifyAll	3038
6.647.3.15	put	3038

6.647.3.16	putAll	3039
6.647.3.17	putAll	3039
6.647.3.18	remove	3039
6.647.3.19	size	3040
6.647.3.20	tryLock	3040
6.647.3.21	unlock	3040
6.647.3.22	values	3040
6.647.3.23	wait	3041
6.647.3.24	wait	3041
6.647.3.25	wait	3042
6.648	decaf::util::StlQueue< T > Class Template Reference	3043
6.648.1	Detailed Description	3044
6.648.2	Constructor & Destructor Documentation	3045
6.648.2.1	StlQueue	3045
6.648.2.2	~StlQueue	3045
6.648.3	Member Function Documentation	3045
6.648.3.1	back	3045
6.648.3.2	back	3045
6.648.3.3	clear	3045
6.648.3.4	empty	3045
6.648.3.5	enqueueFront	3046
6.648.3.6	front	3046
6.648.3.7	front	3046
6.648.3.8	getSafeValue	3046
6.648.3.9	iterator	3046
6.648.3.10	lock	3047
6.648.3.11	notify	3047
6.648.3.12	notifyAll	3047
6.648.3.13	pop	3047
6.648.3.14	push	3048
6.648.3.15	reverse	3048
6.648.3.16	size	3048
6.648.3.17	toArray	3048
6.648.3.18	tryLock	3048
6.648.3.19	unlock	3049
6.648.3.20	wait	3049

6.648.3.21wait	3049
6.648.3.22wait	3050
6.649decaf::util::StlSet< E > Class Template Reference	3051
6.649.1 Detailed Description	3053
6.649.2 Constructor & Destructor Documentation	3053
6.649.2.1 StlSet	3053
6.649.2.2 StlSet	3053
6.649.2.3 StlSet	3053
6.649.2.4 ~StlSet	3053
6.649.3 Member Function Documentation	3053
6.649.3.1 add	3053
6.649.3.2 clear	3054
6.649.3.3 contains	3055
6.649.3.4 copy	3055
6.649.3.5 equals	3055
6.649.3.6 isEmpty	3055
6.649.3.7 iterator	3055
6.649.3.8 iterator	3055
6.649.3.9 remove	3056
6.649.3.10size	3056
6.650activemq::wireformat::stomp::StompCommandConstants Class Reference	3057
6.650.1 Field Documentation	3059
6.650.1.1 ABORT	3059
6.650.1.2 ACK	3059
6.650.1.3 ACK_AUTO	3059
6.650.1.4 ACK_CLIENT	3059
6.650.1.5 ACK_INDIVIDUAL	3059
6.650.1.6 BEGIN	3059
6.650.1.7 BYTES	3059
6.650.1.8 COMMIT	3059
6.650.1.9 CONNECT	3059
6.650.1.10CONNECTED	3059
6.650.1.11DISCONNECT	3059
6.650.1.12ERROR_CMD	3059
6.650.1.13HEADER_ACK	3059
6.650.1.14HEADER_CLIENT_ID	3059

6.650.1.15	HEADER_CONSUMERPRIORITY	3059
6.650.1.16	HEADER_CONTENTLENGTH	3059
6.650.1.17	HEADER_CORRELATIONID	3059
6.650.1.18	HEADER_DESTINATION	3059
6.650.1.19	HEADER_DISPATCH_ASYNC	3059
6.650.1.20	HEADER_EXCLUSIVE	3059
6.650.1.21	HEADER_EXPIRES	3059
6.650.1.22	HEADER_ID	3059
6.650.1.23	HEADER_JMSPRIORITY	3059
6.650.1.24	HEADER_LOGIN	3059
6.650.1.25	HEADER_MAXPENDINGMSGLIMIT	3059
6.650.1.26	HEADER_MESSAGE	3059
6.650.1.27	HEADER_MESSAGEID	3059
6.650.1.28	HEADER_NOLOCAL	3059
6.650.1.29	HEADER_OLDSUBSCRIPTIONNAME	3059
6.650.1.30	HEADER_PASSWORD	3059
6.650.1.31	HEADER_PERSISTENT	3059
6.650.1.32	HEADER_PREFETCHSIZE	3059
6.650.1.33	HEADER_RECEIPT_REQUIRED	3059
6.650.1.34	HEADER_RECEIPTID	3059
6.650.1.35	HEADER_REDELIVERED	3059
6.650.1.36	HEADER_REDELIVERYCOUNT	3059
6.650.1.37	HEADER_REPLYTO	3059
6.650.1.38	HEADER_REQUESTID	3059
6.650.1.39	HEADER_RESPONSEID	3059
6.650.1.40	HEADER_RETROACTIVE	3059
6.650.1.41	HEADER_SELECTOR	3059
6.650.1.42	HEADER_SESSIONID	3059
6.650.1.43	HEADER_SUBSCRIPTION	3059
6.650.1.44	HEADER_SUBSCRIPTIONNAME	3059
6.650.1.45	HEADER_TIMESTAMP	3059
6.650.1.46	HEADER_TRANSACTIONID	3059
6.650.1.47	HEADER_TRANSFORMATION	3059
6.650.1.48	HEADER_TRANSFORMATION_ERROR	3059
6.650.1.49	HEADER_TYPE	3059
6.650.1.50	MESSAGE	3059

6.650.1.51	QUEUE_PREFIX	3059
6.650.1.52	RECEIPT	3059
6.650.1.53	SEND	3059
6.650.1.54	SUBSCRIBE	3059
6.650.1.55	TEMPQUEUE_PREFIX	3059
6.650.1.56	TEMPTOPIC_PREFIX	3059
6.650.1.57	TEXT	3059
6.650.1.58	TOPIC_PREFIX	3059
6.650.1.59	UNSUBSCRIBE	3059
6.651	activemq::wireformat::stomp::StompFrame Class Reference	3061
6.651.1	Detailed Description	3062
6.651.2	Constructor & Destructor Documentation	3062
6.651.2.1	StompFrame	3062
6.651.2.2	~StompFrame	3062
6.651.3	Member Function Documentation	3062
6.651.3.1	clone	3062
6.651.3.2	copy	3062
6.651.3.3	fromStream	3063
6.651.3.4	getBody	3063
6.651.3.5	getBody	3063
6.651.3.6	getBodyLength	3063
6.651.3.7	getCommand	3063
6.651.3.8	getProperties	3063
6.651.3.9	getProperties	3063
6.651.3.10	getProperty	3064
6.651.3.11	hasProperty	3064
6.651.3.12	removeProperty	3064
6.651.3.13	setBody	3064
6.651.3.14	setCommand	3065
6.651.3.15	setProperty	3065
6.651.3.16	oStream	3065
6.652	activemq::wireformat::stomp::StompHelper Class Reference	3066
6.652.1	Detailed Description	3067
6.652.2	Constructor & Destructor Documentation	3067
6.652.2.1	StompHelper	3067
6.652.2.2	~StompHelper	3067

6.652.3 Member Function Documentation	3067
6.652.3.1 convertConsumerId	3067
6.652.3.2 convertConsumerId	3067
6.652.3.3 convertDestination	3067
6.652.3.4 convertDestination	3068
6.652.3.5 convertMessageId	3068
6.652.3.6 convertMessageId	3068
6.652.3.7 convertProducerId	3068
6.652.3.8 convertProducerId	3069
6.652.3.9 convertProperties	3069
6.652.3.10 convertProperties	3069
6.652.3.11 convertTransactionId	3069
6.652.3.12 convertTransactionId	3070
6.653 activemq::wireformat::stomp::StompWireFormat Class Reference	3071
6.653.1 Constructor & Destructor Documentation	3072
6.653.1.1 StompWireFormat	3072
6.653.1.2 ~StompWireFormat	3072
6.653.2 Member Function Documentation	3072
6.653.2.1 createNegotiator	3072
6.653.2.2 getVersion	3072
6.653.2.3 hasNegotiator	3072
6.653.2.4 inReceive	3073
6.653.2.5 marshal	3073
6.653.2.6 setVersion	3073
6.653.2.7 unmarshal	3073
6.654 activemq::wireformat::stomp::StompWireFormatFactory Class Reference	3075
6.654.1 Detailed Description	3075
6.654.2 Constructor & Destructor Documentation	3075
6.654.2.1 StompWireFormatFactory	3075
6.654.2.2 ~StompWireFormatFactory	3075
6.654.3 Member Function Documentation	3075
6.654.3.1 createWireFormat	3075
6.655 cms::Stoppable Class Reference	3076
6.655.1 Detailed Description	3076
6.655.2 Constructor & Destructor Documentation	3076
6.655.2.1 ~Stoppable	3076

6.655.3 Member Function Documentation	3076
6.655.3.1 stop	3076
6.656 decaf::util::logging::StreamHandler Class Reference	3077
6.656.1 Constructor & Destructor Documentation	3078
6.656.1.1 StreamHandler	3078
6.656.1.2 StreamHandler	3078
6.656.1.3 ~StreamHandler	3078
6.656.2 Member Function Documentation	3078
6.656.2.1 close	3078
6.656.2.2 flush	3078
6.656.2.3 getFilter	3078
6.656.2.4 getFormatter	3079
6.656.2.5 getLevel	3079
6.656.2.6 getOutputStream	3079
6.656.2.7 isLoggable	3079
6.656.2.8 publish	3079
6.656.2.9 setFilter	3080
6.656.2.10 setFormatter	3080
6.656.2.11 setLevel	3080
6.657 cms::StreamMessage Class Reference	3081
6.657.1 Detailed Description	3083
6.657.2 Constructor & Destructor Documentation	3084
6.657.2.1 ~StreamMessage	3084
6.657.3 Member Function Documentation	3084
6.657.3.1 readBoolean	3084
6.657.3.2 readByte	3084
6.657.3.3 readBytes	3085
6.657.3.4 readBytes	3085
6.657.3.5 readChar	3086
6.657.3.6 readDouble	3086
6.657.3.7 readFloat	3087
6.657.3.8 readInt	3087
6.657.3.9 readLong	3087
6.657.3.10 readShort	3088
6.657.3.11 readString	3088
6.657.3.12 readUnsignedShort	3089

6.657.3.13	writeBoolean	3089
6.657.3.14	writeByte	3089
6.657.3.15	writeBytes	3090
6.657.3.16	writeBytes	3090
6.657.3.17	writeChar	3090
6.657.3.18	writeDouble	3091
6.657.3.19	writeFloat	3091
6.657.3.20	writeInt	3091
6.657.3.21	writeLong	3092
6.657.3.22	writeShort	3092
6.657.3.23	writeString	3092
6.657.3.24	writeUnsignedShort	3093
6.658	decaf::util::StringTokenizer Class Reference	3094
6.658.1	Constructor & Destructor Documentation	3094
6.658.1.1	StringTokenizer	3094
6.658.1.2	~StringTokenizer	3095
6.658.2	Member Function Documentation	3095
6.658.2.1	countTokens	3095
6.658.2.2	hasMoreTokens	3095
6.658.2.3	nextToken	3095
6.658.2.4	nextToken	3095
6.658.2.5	reset	3096
6.658.2.6	toArray	3096
6.659	activemq::commands::SubscriptionInfo Class Reference	3097
6.659.1	Constructor & Destructor Documentation	3098
6.659.1.1	SubscriptionInfo	3098
6.659.1.2	SubscriptionInfo	3098
6.659.1.3	~SubscriptionInfo	3098
6.659.2	Member Function Documentation	3098
6.659.2.1	cloneDataStructure	3098
6.659.2.2	copyDataStructure	3098
6.659.2.3	equals	3099
6.659.2.4	getClientId	3099
6.659.2.5	getClientId	3099
6.659.2.6	getDataStructureType	3099
6.659.2.7	getDestination	3100

6.659.2.8	getDestination	3100
6.659.2.9	getSelector	3100
6.659.2.10	getSelector	3100
6.659.2.11	getSubscriptionName	3100
6.659.2.12	getSubscriptionName	3100
6.659.2.13	getSubscribedDestination	3100
6.659.2.14	getSubscribedDestination	3100
6.659.2.15	operator=	3100
6.659.2.16	setClientId	3100
6.659.2.17	setDestination	3100
6.659.2.18	setSelector	3100
6.659.2.19	setSubscriptionName	3100
6.659.2.20	setSubscribedDestination	3100
6.659.2.21	toString	3100
6.659.3	Field Documentation	3101
6.659.3.1	clientId	3101
6.659.3.2	destination	3101
6.659.3.3	ID_SUBSCRIPTIONINFO	3101
6.659.3.4	selector	3101
6.659.3.5	subscriptionName	3101
6.659.3.6	subscribedDestination	3101
6.660	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	Class
	Reference	3102
6.660.1	Detailed Description	3102
6.660.2	Constructor & Destructor Documentation	3103
6.660.2.1	SubscriptionInfoMarshaller	3103
6.660.2.2	~SubscriptionInfoMarshaller	3103
6.660.3	Member Function Documentation	3103
6.660.3.1	createObject	3103
6.660.3.2	getDataStructureType	3103
6.660.3.3	looseMarshal	3103
6.660.3.4	looseUnmarshal	3104
6.660.3.5	tightMarshal1	3104
6.660.3.6	tightMarshal2	3104
6.660.3.7	tightUnmarshal	3105
6.661	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	Class
	Reference	3106

6.661.1 Detailed Description	3106
6.661.2 Constructor & Destructor Documentation	3107
6.661.2.1 SubscriptionInfoMarshaller	3107
6.661.2.2 ~SubscriptionInfoMarshaller	3107
6.661.3 Member Function Documentation	3107
6.661.3.1 createObject	3107
6.661.3.2 getDataStructureType	3107
6.661.3.3 looseMarshal	3107
6.661.3.4 looseUnmarshal	3108
6.661.3.5 tightMarshal1	3108
6.661.3.6 tightMarshal2	3108
6.661.3.7 tightUnmarshal	3109
6.662activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class	
Reference	3110
6.662.1 Detailed Description	3110
6.662.2 Constructor & Destructor Documentation	3111
6.662.2.1 SubscriptionInfoMarshaller	3111
6.662.2.2 ~SubscriptionInfoMarshaller	3111
6.662.3 Member Function Documentation	3111
6.662.3.1 createObject	3111
6.662.3.2 getDataStructureType	3111
6.662.3.3 looseMarshal	3111
6.662.3.4 looseUnmarshal	3112
6.662.3.5 tightMarshal1	3112
6.662.3.6 tightMarshal2	3112
6.662.3.7 tightUnmarshal	3113
6.663activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class	
Reference	3114
6.663.1 Detailed Description	3114
6.663.2 Constructor & Destructor Documentation	3115
6.663.2.1 SubscriptionInfoMarshaller	3115
6.663.2.2 ~SubscriptionInfoMarshaller	3115
6.663.3 Member Function Documentation	3115
6.663.3.1 createObject	3115
6.663.3.2 getDataStructureType	3115
6.663.3.3 looseMarshal	3115
6.663.3.4 looseUnmarshal	3116

6.663.3.5	tightMarshal1	3116
6.663.3.6	tightMarshal2	3116
6.663.3.7	tightUnmarshal	3117
6.664	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference	3118
6.664.1	Detailed Description	3118
6.664.2	Constructor & Destructor Documentation	3119
6.664.2.1	SubscriptionInfoMarshaller	3119
6.664.2.2	~SubscriptionInfoMarshaller	3119
6.664.3	Member Function Documentation	3119
6.664.3.1	createObject	3119
6.664.3.2	getDataStructureType	3119
6.664.3.3	looseMarshal	3119
6.664.3.4	looseUnmarshal	3120
6.664.3.5	tightMarshal1	3120
6.664.3.6	tightMarshal2	3120
6.664.3.7	tightUnmarshal	3121
6.665	decaf::util::concurrent::Synchronizable Class Reference	3122
6.665.1	Detailed Description	3123
6.665.2	Constructor & Destructor Documentation	3123
6.665.2.1	~Synchronizable	3123
6.665.3	Member Function Documentation	3123
6.665.3.1	lock	3123
6.665.3.2	notify	3124
6.665.3.3	notifyAll	3125
6.665.3.4	tryLock	3126
6.665.3.5	unlock	3127
6.665.3.6	wait	3128
6.665.3.7	wait	3130
6.665.3.8	wait	3131
6.666	decaf::internal::util::concurrent::SynchronizableImpl Class Reference	3133
6.666.1	Detailed Description	3134
6.666.2	Constructor & Destructor Documentation	3134
6.666.2.1	SynchronizableImpl	3134
6.666.2.2	~SynchronizableImpl	3134
6.666.3	Member Function Documentation	3134
6.666.3.1	lock	3134

6.666.3.2	notify	3134
6.666.3.3	notifyAll	3134
6.666.3.4	tryLock	3135
6.666.3.5	unlock	3135
6.666.3.6	wait	3135
6.666.3.7	wait	3136
6.666.3.8	wait	3136
6.667	activemq::core::Synchronization Class Reference	3137
6.667.1	Detailed Description	3137
6.667.2	Constructor & Destructor Documentation	3137
6.667.2.1	~Synchronization	3137
6.667.3	Member Function Documentation	3137
6.667.3.1	afterCommit	3137
6.667.3.2	afterRollback	3137
6.667.3.3	beforeEnd	3137
6.668	decaf::util::concurrent::SynchronousQueue< E > Class Template Reference	3138
6.668.1	Detailed Description	3139
6.668.2	Constructor & Destructor Documentation	3140
6.668.2.1	SynchronousQueue	3140
6.668.2.2	~SynchronousQueue	3140
6.668.3	Member Function Documentation	3140
6.668.3.1	clear	3140
6.668.3.2	contains	3140
6.668.3.3	containsAll	3140
6.668.3.4	drainTo	3140
6.668.3.5	drainTo	3140
6.668.3.6	equals	3141
6.668.3.7	isEmpty	3141
6.668.3.8	iterator	3141
6.668.3.9	iterator	3141
6.668.3.10	offer	3141
6.668.3.11	offer	3141
6.668.3.12	peek	3142
6.668.3.13	poll	3142
6.668.3.14	poll	3142
6.668.3.15	put	3142

6.668.3.16	remainingCapacity	3143
6.668.3.17	remove	3143
6.668.3.18	removeAll	3143
6.668.3.19	retainAll	3143
6.668.3.20	size	3143
6.668.3.21	take	3143
6.668.3.22	toArray	3144
6.669	decaf::lang::System Class Reference	3145
6.669.1	Constructor & Destructor Documentation	3145
6.669.1.1	System	3145
6.669.1.2	~System	3145
6.669.2	Member Function Documentation	3145
6.669.2.1	availableProcessors	3145
6.669.2.2	currentTimeMillis	3146
6.669.2.3	getenv	3146
6.669.2.4	getenv	3146
6.669.2.5	nanoTime	3147
6.669.2.6	setenv	3147
6.669.2.7	unsetenv	3147
6.670	activemq::threads::Task Class Reference	3148
6.670.1	Detailed Description	3148
6.670.2	Constructor & Destructor Documentation	3148
6.670.2.1	~Task	3148
6.670.3	Member Function Documentation	3148
6.670.3.1	iterate	3148
6.671	decaf::util::concurrent::TaskListener Class Reference	3149
6.671.1	Constructor & Destructor Documentation	3149
6.671.1.1	~TaskListener	3149
6.671.2	Member Function Documentation	3149
6.671.2.1	onTaskComplete	3149
6.671.2.2	onTaskException	3149
6.672	activemq::threads::TaskRunner Class Reference	3150
6.672.1	Constructor & Destructor Documentation	3150
6.672.1.1	~TaskRunner	3150
6.672.2	Member Function Documentation	3150
6.672.2.1	shutdown	3150

6.672.2.2 shutdown	3150
6.672.2.3 wakeup	3151
6.673decaf::net::TcpSocket Class Reference	3152
6.673.1 Detailed Description	3153
6.673.2 Constructor & Destructor Documentation	3153
6.673.2.1 TcpSocket	3153
6.673.2.2 TcpSocket	3154
6.673.2.3 ~TcpSocket	3154
6.673.3 Member Function Documentation	3154
6.673.3.1 checkResult	3154
6.673.3.2 close	3154
6.673.3.3 connect	3154
6.673.3.4 connect	3154
6.673.3.5 getInputStream	3155
6.673.3.6 getKeepAlive	3155
6.673.3.7 getOutputStream	3155
6.673.3.8 getReceiveBufferSize	3155
6.673.3.9 getReuseAddress	3156
6.673.3.10getSendBufferSize	3156
6.673.3.11getSocketHandle	3156
6.673.3.12getSoLinger	3156
6.673.3.13getSoTimeout	3157
6.673.3.14getTcpNoDelay	3157
6.673.3.15sConnected	3157
6.673.3.16setKeepAlive	3157
6.673.3.17setReceiveBufferSize	3158
6.673.3.18setReuseAddress	3158
6.673.3.19setSendBufferSize	3158
6.673.3.20setSoLinger	3158
6.673.3.21setSoTimeout	3159
6.673.3.22setTcpNoDelay	3159
6.674activemq::transport::tcp::TcpTransport Class Reference	3160
6.674.1 Detailed Description	3160
6.674.2 Constructor & Destructor Documentation	3160
6.674.2.1 TcpTransport	3160
6.674.2.2 TcpTransport	3161

6.674.2.3 ~TcpTransport	3161
6.674.3 Member Function Documentation	3161
6.674.3.1 close	3161
6.674.3.2 isClosed	3161
6.674.3.3 isConnected	3161
6.674.3.4 isFaultTolerant	3162
6.675activemq::transport::tcp::TcpTransportFactory Class Reference	3163
6.675.1 Detailed Description	3163
6.675.2 Constructor & Destructor Documentation	3164
6.675.2.1 ~TcpTransportFactory	3164
6.675.3 Member Function Documentation	3164
6.675.3.1 create	3164
6.675.3.2 createComposite	3164
6.675.3.3 doCreateComposite	3164
6.676cms::TemporaryQueue Class Reference	3166
6.676.1 Detailed Description	3166
6.676.2 Constructor & Destructor Documentation	3166
6.676.2.1 ~TemporaryQueue	3166
6.676.3 Member Function Documentation	3166
6.676.3.1 destroy	3166
6.676.3.2 getQueueName	3167
6.677cms::TemporaryTopic Class Reference	3168
6.677.1 Detailed Description	3168
6.677.2 Constructor & Destructor Documentation	3168
6.677.2.1 ~TemporaryTopic	3168
6.677.3 Member Function Documentation	3168
6.677.3.1 destroy	3168
6.677.3.2 getTopicName	3169
6.678cms::TextMessage Class Reference	3170
6.678.1 Detailed Description	3170
6.678.2 Constructor & Destructor Documentation	3170
6.678.2.1 ~TextMessage	3170
6.678.3 Member Function Documentation	3170
6.678.3.1 getText	3170
6.678.3.2 setText	3171
6.678.3.3 setText	3171

6.679	decaf::util::concurrent::ThreadFactory Class Reference	3172
6.679.1	Detailed Description	3172
6.679.2	Constructor & Destructor Documentation	3172
6.679.2.1	~ThreadFactory	3172
6.679.3	Member Function Documentation	3172
6.679.3.1	newThread	3172
6.680	decaf::lang::ThreadGroup Class Reference	3174
6.680.1	Detailed Description	3174
6.680.2	Constructor & Destructor Documentation	3174
6.680.2.1	ThreadGroup	3174
6.680.2.2	~ThreadGroup	3174
6.681	decaf::util::concurrent::ThreadPool Class Reference	3175
6.681.1	Detailed Description	3176
6.681.2	Member Typedef Documentation	3177
6.681.2.1	Task	3177
6.681.3	Constructor & Destructor Documentation	3177
6.681.3.1	ThreadPool	3177
6.681.3.2	~ThreadPool	3177
6.681.4	Member Function Documentation	3177
6.681.4.1	deQueueTask	3177
6.681.4.2	getBacklog	3177
6.681.4.3	getBlockSize	3177
6.681.4.4	getFreeThreadCount	3177
6.681.4.5	getInstance	3178
6.681.4.6	getMaxThreads	3178
6.681.4.7	getPoolSize	3178
6.681.4.8	onTaskCompleted	3178
6.681.4.9	onTaskException	3178
6.681.4.10	onTaskStarted	3179
6.681.4.11	queueTask	3179
6.681.4.12	reserve	3179
6.681.4.13	setBlockSize	3179
6.681.4.14	setMaxThreads	3180
6.681.5	Field Documentation	3180
6.681.5.1	DEFAULT_MAX_BLOCK_SIZE	3180
6.681.5.2	DEFAULT_MAX_POOL_SIZE	3180

6.682decaf::lang::Throwable Class Reference	3181
6.682.1 Detailed Description	3181
6.682.2 Constructor & Destructor Documentation	3182
6.682.2.1 Throwable	3182
6.682.2.2 ~Throwable	3182
6.682.3 Member Function Documentation	3182
6.682.3.1 clone	3182
6.682.3.2 getCause	3183
6.682.3.3 getMessage	3183
6.682.3.4 getStackTrace	3183
6.682.3.5 getStackTraceString	3183
6.682.3.6 initCause	3183
6.682.3.7 printStackTrace	3184
6.682.3.8 printStackTrace	3184
6.682.3.9 setMark	3184
6.683decaf::util::concurrent::TimeoutException Class Reference	3185
6.683.1 Constructor & Destructor Documentation	3185
6.683.1.1 TimeoutException	3185
6.683.1.2 TimeoutException	3185
6.683.1.3 TimeoutException	3186
6.683.1.4 TimeoutException	3186
6.683.1.5 TimeoutException	3186
6.683.1.6 TimeoutException	3186
6.683.1.7 ~TimeoutException	3187
6.683.2 Member Function Documentation	3187
6.683.2.1 clone	3187
6.684decaf::util::Timer Class Reference	3188
6.684.1 Detailed Description	3189
6.684.2 Constructor & Destructor Documentation	3190
6.684.2.1 Timer	3190
6.684.2.2 ~Timer	3190
6.684.3 Member Function Documentation	3190
6.684.3.1 cancel	3190
6.684.3.2 purge	3190
6.684.3.3 schedule	3190
6.684.3.4 schedule	3191

6.684.3.5	schedule	3192
6.684.3.6	schedule	3192
6.684.3.7	schedule	3193
6.684.3.8	schedule	3194
6.684.3.9	schedule	3194
6.684.3.10	schedule	3195
6.684.3.11	scheduleAtFixedRate	3195
6.684.3.12	scheduleAtFixedRate	3196
6.684.3.13	scheduleAtFixedRate	3197
6.684.3.14	scheduleAtFixedRate	3197
6.685	decaf::util::TimerTask Class Reference	3199
6.685.1	Detailed Description	3199
6.685.2	Constructor & Destructor Documentation	3200
6.685.2.1	TimerTask	3200
6.685.2.2	~TimerTask	3200
6.685.3	Member Function Documentation	3200
6.685.3.1	cancel	3200
6.685.3.2	getWhen	3200
6.685.3.3	isScheduled	3200
6.685.3.4	scheduledExecutionTime	3200
6.685.3.5	setScheduledTime	3201
6.685.4	Friends And Related Function Documentation	3201
6.685.4.1	decaf::internal::util::TimerTaskHeap	3201
6.685.4.2	Timer	3201
6.685.4.3	TimerImpl	3201
6.686	decaf::internal::util::TimerTaskHeap Class Reference	3202
6.686.1	Detailed Description	3202
6.686.2	Constructor & Destructor Documentation	3203
6.686.2.1	TimerTaskHeap	3203
6.686.2.2	~TimerTaskHeap	3203
6.686.3	Member Function Documentation	3203
6.686.3.1	adjustMinimum	3203
6.686.3.2	deleteIfCancelled	3203
6.686.3.3	find	3203
6.686.3.4	insert	3203
6.686.3.5	isEmpty	3203

6.686.3.6 peek	3204
6.686.3.7 remove	3204
6.686.3.8 reset	3204
6.686.3.9 size	3204
6.687decaf::util::concurrent::TimeUnit Class Reference	3205
6.687.1 Detailed Description	3206
6.687.2 Constructor & Destructor Documentation	3207
6.687.2.1 TimeUnit	3207
6.687.2.2 ~TimeUnit	3207
6.687.3 Member Function Documentation	3207
6.687.3.1 compareTo	3207
6.687.3.2 convert	3208
6.687.3.3 equals	3208
6.687.3.4 operator<	3208
6.687.3.5 operator==	3209
6.687.3.6 sleep	3209
6.687.3.7 timedJoin	3209
6.687.3.8 timedWait	3210
6.687.3.9 toDays	3210
6.687.3.10toHours	3211
6.687.3.11toMicros	3211
6.687.3.12toMillis	3211
6.687.3.13toMinutes	3212
6.687.3.14toNanos	3212
6.687.3.15toSeconds	3212
6.687.3.16toString	3213
6.687.3.17valueOf	3213
6.687.4 Field Documentation	3213
6.687.4.1 DAYS	3213
6.687.4.2 HOURS	3213
6.687.4.3 MICROSECONDS	3213
6.687.4.4 MILLISECONDS	3213
6.687.4.5 MINUTES	3213
6.687.4.6 NANOSECONDS	3213
6.687.4.7 SECONDS	3214
6.687.4.8 values	3214

6.688	cms::Topic Class Reference	3215
6.688.1	Detailed Description	3215
6.688.2	Constructor & Destructor Documentation	3215
6.688.2.1	~Topic	3215
6.688.3	Member Function Documentation	3215
6.688.3.1	getTopicName	3215
6.689	activemq::state::Tracked Class Reference	3216
6.689.1	Constructor & Destructor Documentation	3216
6.689.1.1	Tracked	3216
6.689.1.2	Tracked	3216
6.689.1.3	~Tracked	3216
6.689.2	Member Function Documentation	3216
6.689.2.1	isWaitingForResponse	3216
6.689.2.2	onResponse	3216
6.690	activemq::commands::TransactionId Class Reference	3217
6.690.1	Member Typedef Documentation	3217
6.690.1.1	COMPARATOR	3217
6.690.2	Constructor & Destructor Documentation	3218
6.690.2.1	TransactionId	3218
6.690.2.2	TransactionId	3218
6.690.2.3	~TransactionId	3218
6.690.3	Member Function Documentation	3218
6.690.3.1	cloneDataStructure	3218
6.690.3.2	compareTo	3218
6.690.3.3	copyDataStructure	3218
6.690.3.4	equals	3218
6.690.3.5	equals	3218
6.690.3.6	getDataStructureType	3219
6.690.3.7	operator<	3219
6.690.3.8	operator=	3219
6.690.3.9	operator==	3219
6.690.3.10	toString	3219
6.690.4	Field Documentation	3219
6.690.4.1	ID_TRANSACTIONID	3219
6.691	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference	3220
6.691.1	Detailed Description	3220

6.691.2 Constructor & Destructor Documentation	3221
6.691.2.1 TransactionIdMarshaller	3221
6.691.2.2 ~TransactionIdMarshaller	3221
6.691.3 Member Function Documentation	3221
6.691.3.1 looseMarshal	3221
6.691.3.2 looseUnmarshal	3221
6.691.3.3 tightMarshal1	3222
6.691.3.4 tightMarshal2	3222
6.691.3.5 tightUnmarshal	3223
6.692activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Refer- ence	3224
6.692.1 Detailed Description	3224
6.692.2 Constructor & Destructor Documentation	3225
6.692.2.1 TransactionIdMarshaller	3225
6.692.2.2 ~TransactionIdMarshaller	3225
6.692.3 Member Function Documentation	3225
6.692.3.1 looseMarshal	3225
6.692.3.2 looseUnmarshal	3225
6.692.3.3 tightMarshal1	3226
6.692.3.4 tightMarshal2	3226
6.692.3.5 tightUnmarshal	3227
6.693activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Refer- ence	3228
6.693.1 Detailed Description	3228
6.693.2 Constructor & Destructor Documentation	3229
6.693.2.1 TransactionIdMarshaller	3229
6.693.2.2 ~TransactionIdMarshaller	3229
6.693.3 Member Function Documentation	3229
6.693.3.1 looseMarshal	3229
6.693.3.2 looseUnmarshal	3229
6.693.3.3 tightMarshal1	3230
6.693.3.4 tightMarshal2	3230
6.693.3.5 tightUnmarshal	3231
6.694activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Refer- ence	3232
6.694.1 Detailed Description	3232
6.694.2 Constructor & Destructor Documentation	3233

6.694.2.1 TransactionIdMarshaller	3233
6.694.2.2 ~TransactionIdMarshaller	3233
6.694.3 Member Function Documentation	3233
6.694.3.1 looseMarshal	3233
6.694.3.2 looseUnmarshal	3233
6.694.3.3 tightMarshal1	3234
6.694.3.4 tightMarshal2	3234
6.694.3.5 tightUnmarshal	3235
6.695activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference	3236
6.695.1 Detailed Description	3236
6.695.2 Constructor & Destructor Documentation	3237
6.695.2.1 TransactionIdMarshaller	3237
6.695.2.2 ~TransactionIdMarshaller	3237
6.695.3 Member Function Documentation	3237
6.695.3.1 looseMarshal	3237
6.695.3.2 looseUnmarshal	3237
6.695.3.3 tightMarshal1	3238
6.695.3.4 tightMarshal2	3238
6.695.3.5 tightUnmarshal	3239
6.696activemq::commands::TransactionInfo Class Reference	3240
6.696.1 Constructor & Destructor Documentation	3241
6.696.1.1 TransactionInfo	3241
6.696.1.2 TransactionInfo	3241
6.696.1.3 ~TransactionInfo	3241
6.696.2 Member Function Documentation	3241
6.696.2.1 cloneDataStructure	3241
6.696.2.2 copyDataStructure	3241
6.696.2.3 equals	3241
6.696.2.4 getConnectionId	3242
6.696.2.5 getConnectionId	3242
6.696.2.6 getDataStructureType	3242
6.696.2.7 getTransactionId	3242
6.696.2.8 getTransactionId	3242
6.696.2.9 getType	3242
6.696.2.10sTransactionInfo	3242
6.696.2.11operator=	3243

6.696.2.12	setConnectionId	3243
6.696.2.13	setTransactionId	3243
6.696.2.14	setType	3243
6.696.2.15	toString	3243
6.696.2.16	visit	3243
6.696.3	Field Documentation	3243
6.696.3.1	connectionId	3243
6.696.3.2	ID_TRANSACTIONINFO	3243
6.696.3.3	transactionId	3243
6.696.3.4	type	3243
6.697	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference	3245
6.697.1	Detailed Description	3245
6.697.2	Constructor & Destructor Documentation	3246
6.697.2.1	TransactionInfoMarshaller	3246
6.697.2.2	~TransactionInfoMarshaller	3246
6.697.3	Member Function Documentation	3246
6.697.3.1	createObject	3246
6.697.3.2	getDataStructureType	3246
6.697.3.3	looseMarshal	3246
6.697.3.4	looseUnmarshal	3247
6.697.3.5	tightMarshal1	3247
6.697.3.6	tightMarshal2	3247
6.697.3.7	tightUnmarshal	3248
6.698	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference	3249
6.698.1	Detailed Description	3249
6.698.2	Constructor & Destructor Documentation	3250
6.698.2.1	TransactionInfoMarshaller	3250
6.698.2.2	~TransactionInfoMarshaller	3250
6.698.3	Member Function Documentation	3250
6.698.3.1	createObject	3250
6.698.3.2	getDataStructureType	3250
6.698.3.3	looseMarshal	3250
6.698.3.4	looseUnmarshal	3251
6.698.3.5	tightMarshal1	3251
6.698.3.6	tightMarshal2	3251

6.698.3.7 tightUnmarshal	3252
6.699activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference	3253
6.699.1 Detailed Description	3253
6.699.2 Constructor & Destructor Documentation	3254
6.699.2.1 TransactionInfoMarshaller	3254
6.699.2.2 ~TransactionInfoMarshaller	3254
6.699.3 Member Function Documentation	3254
6.699.3.1 createObject	3254
6.699.3.2 getDataStructureType	3254
6.699.3.3 looseMarshal	3254
6.699.3.4 looseUnmarshal	3255
6.699.3.5 tightMarshal1	3255
6.699.3.6 tightMarshal2	3255
6.699.3.7 tightUnmarshal	3256
6.700activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference	3257
6.700.1 Detailed Description	3257
6.700.2 Constructor & Destructor Documentation	3258
6.700.2.1 TransactionInfoMarshaller	3258
6.700.2.2 ~TransactionInfoMarshaller	3258
6.700.3 Member Function Documentation	3258
6.700.3.1 createObject	3258
6.700.3.2 getDataStructureType	3258
6.700.3.3 looseMarshal	3258
6.700.3.4 looseUnmarshal	3259
6.700.3.5 tightMarshal1	3259
6.700.3.6 tightMarshal2	3259
6.700.3.7 tightUnmarshal	3260
6.701activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference	3261
6.701.1 Detailed Description	3261
6.701.2 Constructor & Destructor Documentation	3262
6.701.2.1 TransactionInfoMarshaller	3262
6.701.2.2 ~TransactionInfoMarshaller	3262
6.701.3 Member Function Documentation	3262
6.701.3.1 createObject	3262

6.701.3.2	getDataStructureType	3262
6.701.3.3	looseMarshal	3262
6.701.3.4	looseUnmarshal	3263
6.701.3.5	tightMarshal1	3263
6.701.3.6	tightMarshal2	3263
6.701.3.7	tightUnmarshal	3264
6.702	activemq::state::TransactionState Class Reference	3265
6.702.1	Constructor & Destructor Documentation	3266
6.702.1.1	TransactionState	3266
6.702.1.2	~TransactionState	3266
6.702.2	Member Function Documentation	3266
6.702.2.1	addCommand	3266
6.702.2.2	checkShutdown	3266
6.702.2.3	getCommands	3266
6.702.2.4	getId	3266
6.702.2.5	getPreparedResult	3266
6.702.2.6	isPrepared	3266
6.702.2.7	setPrepared	3266
6.702.2.8	setPreparedResult	3266
6.702.2.9	shutdown	3266
6.702.2.10	toString	3266
6.703	decaf::internal::util::concurrent::Transferer< E > Class Template Reference	3267
6.703.1	Detailed Description	3267
6.704	decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference	3268
6.704.1	Detailed Description	3268
6.704.2	Constructor & Destructor Documentation	3268
6.704.2.1	TransferQueue	3268
6.704.2.2	~TransferQueue	3269
6.704.3	Member Function Documentation	3269
6.704.3.1	transfer	3269
6.704.3.2	transfer	3269
6.705	decaf::internal::util::concurrent::TransferStack< E > Class Template Reference	3271
6.705.1	Constructor & Destructor Documentation	3271
6.705.1.1	TransferStack	3271
6.705.1.2	~TransferStack	3271
6.705.2	Member Function Documentation	3271

6.705.2.1 transfer	3271
6.705.2.2 transfer	3272
6.706activemq::transport::Transport Class Reference	3273
6.706.1 Detailed Description	3274
6.706.2 Constructor & Destructor Documentation	3274
6.706.2.1 ~Transport	3274
6.706.3 Member Function Documentation	3274
6.706.3.1 getRemoteAddress	3274
6.706.3.2 getTransportListener	3274
6.706.3.3 isClosed	3275
6.706.3.4 isConnected	3275
6.706.3.5 isFaultTolerant	3275
6.706.3.6 narrow	3275
6.706.3.7 oneway	3276
6.706.3.8 reconnect	3276
6.706.3.9 request	3276
6.706.3.10request	3277
6.706.3.11setTransportListener	3277
6.706.3.12setWireFormat	3278
6.706.3.13start	3278
6.706.3.14stop	3278
6.707activemq::transport::TransportFactory Class Reference	3279
6.707.1 Detailed Description	3279
6.707.2 Constructor & Destructor Documentation	3279
6.707.2.1 ~TransportFactory	3279
6.707.3 Member Function Documentation	3279
6.707.3.1 create	3279
6.707.3.2 createComposite	3280
6.708activemq::transport::TransportFilter Class Reference	3281
6.708.1 Detailed Description	3282
6.708.2 Constructor & Destructor Documentation	3283
6.708.2.1 TransportFilter	3283
6.708.2.2 ~TransportFilter	3283
6.708.3 Member Function Documentation	3283
6.708.3.1 close	3283
6.708.3.2 fire	3283

6.708.3.3	fire	3283
6.708.3.4	getRemoteAddress	3284
6.708.3.5	getTransportListener	3284
6.708.3.6	isClosed	3284
6.708.3.7	isConnected	3284
6.708.3.8	isFaultTolerant	3284
6.708.3.9	narrow	3285
6.708.3.10	onCommand	3285
6.708.3.11	oneway	3285
6.708.3.12	onException	3286
6.708.3.13	reconnect	3286
6.708.3.14	request	3286
6.708.3.15	request	3287
6.708.3.16	setTransportListener	3287
6.708.3.17	setWireFormat	3287
6.708.3.18	start	3287
6.708.3.19	stop	3288
6.708.3.20	transportInterrupted	3288
6.708.3.21	transportResumed	3288
6.708.4	Field Documentation	3288
6.708.4.1	listener	3288
6.708.4.2	next	3288
6.709	activemq::transport::TransportListener Class Reference	3289
6.709.1	Detailed Description	3289
6.709.2	Constructor & Destructor Documentation	3289
6.709.2.1	~TransportListener	3289
6.709.3	Member Function Documentation	3289
6.709.3.1	onCommand	3289
6.709.3.2	onException	3290
6.709.3.3	transportInterrupted	3290
6.709.3.4	transportResumed	3290
6.710	activemq::transport::TransportRegistry Class Reference	3291
6.710.1	Detailed Description	3291
6.710.2	Constructor & Destructor Documentation	3292
6.710.2.1	~TransportRegistry	3292
6.710.3	Member Function Documentation	3292

6.710.3.1 findFactory	3292
6.710.3.2 getInstance	3292
6.710.3.3 getTransportNames	3292
6.710.3.4 registerFactory	3292
6.710.3.5 unregisterFactory	3293
6.711decaf::net::UnknownHostException Class Reference	3294
6.711.1 Constructor & Destructor Documentation	3294
6.711.1.1 UnknownHostException	3294
6.711.1.2 UnknownHostException	3294
6.711.1.3 UnknownHostException	3295
6.711.1.4 UnknownHostException	3295
6.711.1.5 UnknownHostException	3295
6.711.1.6 UnknownHostException	3295
6.711.1.7 ~UnknownHostException	3296
6.711.2 Member Function Documentation	3296
6.711.2.1 clone	3296
6.712decaf::net::UnknownServiceException Class Reference	3297
6.712.1 Constructor & Destructor Documentation	3297
6.712.1.1 UnknownServiceException	3297
6.712.1.2 UnknownServiceException	3297
6.712.1.3 UnknownServiceException	3298
6.712.1.4 UnknownServiceException	3298
6.712.1.5 UnknownServiceException	3298
6.712.1.6 UnknownServiceException	3298
6.712.1.7 ~UnknownServiceException	3299
6.712.2 Member Function Documentation	3299
6.712.2.1 clone	3299
6.713decaf::io::UnsupportedEncodingException Class Reference	3300
6.713.1 Detailed Description	3300
6.713.2 Constructor & Destructor Documentation	3300
6.713.2.1 UnsupportedEncodingException	3300
6.713.2.2 UnsupportedEncodingException	3301
6.713.2.3 UnsupportedEncodingException	3301
6.713.2.4 UnsupportedEncodingException	3301
6.713.2.5 UnsupportedEncodingException	3301
6.713.2.6 UnsupportedEncodingException	3301

6.713.2.7 ~UnsupportedEncodingException	3302
6.713.3 Member Function Documentation	3302
6.713.3.1 clone	3302
6.714decaf::lang::exceptions::UnsupportedOperationException Class Reference	3303
6.714.1 Constructor & Destructor Documentation	3303
6.714.1.1 UnsupportedOperationException	3303
6.714.1.2 UnsupportedOperationException	3303
6.714.1.3 UnsupportedOperationException	3304
6.714.1.4 UnsupportedOperationException	3304
6.714.1.5 UnsupportedOperationException	3304
6.714.1.6 UnsupportedOperationException	3304
6.714.1.7 ~UnsupportedOperationException	3305
6.714.2 Member Function Documentation	3305
6.714.2.1 clone	3305
6.715cms::UnsupportedOperationException Class Reference	3306
6.715.1 Detailed Description	3306
6.715.2 Constructor & Destructor Documentation	3306
6.715.2.1 UnsupportedOperationException	3306
6.715.2.2 UnsupportedOperationException	3306
6.715.2.3 UnsupportedOperationException	3306
6.715.2.4 UnsupportedOperationException	3306
6.715.2.5 ~UnsupportedOperationException	3306
6.716decaf::net::URI Class Reference	3308
6.716.1 Detailed Description	3310
6.716.2 Constructor & Destructor Documentation	3310
6.716.2.1 URI	3310
6.716.2.2 URI	3310
6.716.2.3 URI	3310
6.716.2.4 URI	3310
6.716.2.5 URI	3311
6.716.2.6 URI	3311
6.716.2.7 URI	3311
6.716.2.8 ~URI	3312
6.716.3 Member Function Documentation	3312
6.716.3.1 compareTo	3312
6.716.3.2 create	3312

6.716.3.3 equals	3312
6.716.3.4 getAuthority	3312
6.716.3.5 getFragment	3312
6.716.3.6 getHost	3313
6.716.3.7 getPath	3313
6.716.3.8 getPort	3313
6.716.3.9 getQuery	3313
6.716.3.10 getRawAuthority	3313
6.716.3.11 getRawFragment	3313
6.716.3.12 getRawPath	3313
6.716.3.13 getRawQuery	3314
6.716.3.14 getRawSchemeSpecificPart	3314
6.716.3.15 getRawUserInfo	3314
6.716.3.16 getScheme	3314
6.716.3.17 getSchemeSpecificPart	3314
6.716.3.18 getUserInfo	3315
6.716.3.19 isAbsolute	3315
6.716.3.20 isOpaque	3315
6.716.3.21 normalize	3315
6.716.3.22 operator<	3315
6.716.3.23 operator==	3316
6.716.3.24 parseServerAuthority	3316
6.716.3.25 relativize	3316
6.716.3.26 resolve	3317
6.716.3.27 resolve	3317
6.716.3.28 toString	3318
6.716.3.29 toURL	3318
6.717.decaf::internal::net::URLEncoderDecoder Class Reference	3319
6.717.1 Constructor & Destructor Documentation	3319
6.717.1.1 URLEncoderDecoder	3319
6.717.1.2 ~URLEncoderDecoder	3319
6.717.2 Member Function Documentation	3319
6.717.2.1 decode	3319
6.717.2.2 encodeOthers	3320
6.717.2.3 quoteIllegal	3320
6.717.2.4 validate	3320

6.717.2.5 validateSimple	3321
6.718decaf::internal::net::URIHelper Class Reference	3322
6.718.1 Detailed Description	3323
6.718.2 Constructor & Destructor Documentation	3323
6.718.2.1 URIHelper	3323
6.718.2.2 URIHelper	3323
6.718.2.3 ~URIHelper	3324
6.718.3 Member Function Documentation	3324
6.718.3.1 isValidDomainName	3324
6.718.3.2 isValidHexChar	3324
6.718.3.3 isValidHost	3324
6.718.3.4 isValidIP4Word	3324
6.718.3.5 isValidIP6Address	3325
6.718.3.6 isValidIPv4Address	3325
6.718.3.7 parseAuthority	3325
6.718.3.8 parseURI	3326
6.718.3.9 validateAuthority	3326
6.718.3.10validateFragment	3326
6.718.3.11validatePath	3327
6.718.3.12validateQuery	3327
6.718.3.13validateScheme	3327
6.718.3.14validateSsp	3328
6.718.3.15validateUserinfo	3328
6.719activemq::transport::failover::URIPool Class Reference	3329
6.719.1 Constructor & Destructor Documentation	3329
6.719.1.1 URIPool	3329
6.719.1.2 URIPool	3329
6.719.1.3 ~URIPool	3330
6.719.2 Member Function Documentation	3330
6.719.2.1 addURI	3330
6.719.2.2 addURIs	3330
6.719.2.3 getURI	3330
6.719.2.4 isRandomize	3330
6.719.2.5 removeURI	3330
6.719.2.6 setRandomize	3331
6.720activemq::util::URISupport Class Reference	3332

6.720.1 Member Function Documentation	3332
6.720.1.1 createQueryString	3332
6.720.1.2 parseComposite	3333
6.720.1.3 parseQuery	3333
6.720.1.4 parseQuery	3333
6.720.1.5 parseURL	3334
6.721decaf::net::URISyntaxException Class Reference	3335
6.721.1 Constructor & Destructor Documentation	3335
6.721.1.1 URISyntaxException	3335
6.721.1.2 URISyntaxException	3336
6.721.1.3 URISyntaxException	3336
6.721.1.4 URISyntaxException	3336
6.721.1.5 URISyntaxException	3336
6.721.1.6 URISyntaxException	3336
6.721.1.7 URISyntaxException	3337
6.721.1.8 URISyntaxException	3337
6.721.1.9 ~URISyntaxException	3337
6.721.2 Member Function Documentation	3337
6.721.2.1 clone	3337
6.721.2.2 getIndex	3338
6.721.2.3 getInput	3338
6.721.2.4 getReason	3338
6.722decaf::internal::net::URIType Class Reference	3339
6.722.1 Detailed Description	3341
6.722.2 Constructor & Destructor Documentation	3341
6.722.2.1 URIType	3341
6.722.2.2 URIType	3341
6.722.2.3 ~URIType	3341
6.722.3 Member Function Documentation	3341
6.722.3.1 getAuthority	3341
6.722.3.2 getFragment	3341
6.722.3.3 getHost	3341
6.722.3.4 getPath	3341
6.722.3.5 getPort	3342
6.722.3.6 getQuery	3342
6.722.3.7 getScheme	3342

6.722.3.8	getSchemeSpecificPart	3342
6.722.3.9	getSource	3342
6.722.3.10	getUserInfo	3342
6.722.3.11	isAbsolute	3343
6.722.3.12	isOpaque	3343
6.722.3.13	isServerAuthority	3343
6.722.3.14	isValid	3343
6.722.3.15	setAbsolute	3343
6.722.3.16	setAuthority	3343
6.722.3.17	setFragment	3344
6.722.3.18	setHost	3344
6.722.3.19	setOpaque	3344
6.722.3.20	setPath	3344
6.722.3.21	setPort	3344
6.722.3.22	setQuery	3344
6.722.3.23	setScheme	3345
6.722.3.24	setSchemeSpecificPart	3345
6.722.3.25	setServerAuthority	3345
6.722.3.26	setSource	3345
6.722.3.27	setUserInfo	3345
6.722.3.28	setValid	3345
6.723	decaf::net::URL Class Reference	3347
6.723.1	Detailed Description	3347
6.723.2	Constructor & Destructor Documentation	3348
6.723.2.1	URL	3348
6.723.2.2	URL	3348
6.723.2.3	~URL	3348
6.724	decaf::net::URLDecoder Class Reference	3349
6.724.1	Constructor & Destructor Documentation	3349
6.724.1.1	~URLDecoder	3349
6.724.2	Member Function Documentation	3349
6.724.2.1	decode	3349
6.725	decaf::net::URLEncoder Class Reference	3350
6.725.1	Constructor & Destructor Documentation	3350
6.725.1.1	~URLEncoder	3350
6.725.2	Member Function Documentation	3350

6.725.2.1 encode	3350
6.726activemq::util::Usage Class Reference	3351
6.726.1 Constructor & Destructor Documentation	3351
6.726.1.1 ~Usage	3351
6.726.2 Member Function Documentation	3351
6.726.2.1 decreaseUsage	3351
6.726.2.2 enqueueUsage	3351
6.726.2.3 increaseUsage	3352
6.726.2.4 isFull	3352
6.726.2.5 waitForSpace	3352
6.726.2.6 waitForSpace	3352
6.727decaf::io::UTFDataFormatException Class Reference	3353
6.727.1 Detailed Description	3353
6.727.2 Constructor & Destructor Documentation	3353
6.727.2.1 UTFDataFormatException	3353
6.727.2.2 UTFDataFormatException	3354
6.727.2.3 UTFDataFormatException	3354
6.727.2.4 UTFDataFormatException	3354
6.727.2.5 UTFDataFormatException	3354
6.727.2.6 UTFDataFormatException	3354
6.727.2.7 ~UTFDataFormatException	3355
6.727.3 Member Function Documentation	3355
6.727.3.1 clone	3355
6.728decaf::util::UUID Class Reference	3356
6.728.1 Detailed Description	3357
6.728.2 Constructor & Destructor Documentation	3357
6.728.2.1 UUID	3357
6.728.2.2 ~UUID	3358
6.728.3 Member Function Documentation	3358
6.728.3.1 clockSequence	3358
6.728.3.2 compareTo	3358
6.728.3.3 equals	3358
6.728.3.4 fromString	3359
6.728.3.5 getLeastSignificantBits	3359
6.728.3.6 getMostSignificantBits	3359
6.728.3.7 nameUUIDFromBytes	3359

6.728.3.8 nameUUIDFromBytes	3359
6.728.3.9 node	3360
6.728.3.10 operator<	3360
6.728.3.11 operator==	3360
6.728.3.12 randomUUID	3360
6.728.3.13 timestamp	3361
6.728.3.14 toString	3361
6.728.3.15 variant	3361
6.728.3.16 version	3361
6.729 activemq::wireformat::WireFormat Class Reference	3363
6.729.1 Detailed Description	3363
6.729.2 Constructor & Destructor Documentation	3364
6.729.2.1 ~WireFormat	3364
6.729.3 Member Function Documentation	3364
6.729.3.1 createNegotiator	3364
6.729.3.2 getVersion	3364
6.729.3.3 hasNegotiator	3364
6.729.3.4 inReceive	3365
6.729.3.5 marshal	3365
6.729.3.6 setVersion	3365
6.729.3.7 unmarshal	3366
6.730 activemq::wireformat::WireFormatFactory Class Reference	3367
6.730.1 Detailed Description	3367
6.730.2 Constructor & Destructor Documentation	3367
6.730.2.1 ~WireFormatFactory	3367
6.730.3 Member Function Documentation	3367
6.730.3.1 createWireFormat	3367
6.731 activemq::commands::WireFormatInfo Class Reference	3369
6.731.1 Constructor & Destructor Documentation	3371
6.731.1.1 WireFormatInfo	3371
6.731.1.2 ~WireFormatInfo	3371
6.731.2 Member Function Documentation	3371
6.731.2.1 afterUnmarshal	3371
6.731.2.2 beforeMarshal	3372
6.731.2.3 cloneDataStructure	3372
6.731.2.4 copyDataStructure	3372

6.731.2.5 equals	3372
6.731.2.6 getCacheSize	3373
6.731.2.7 getDataStructureType	3373
6.731.2.8 getMagic	3373
6.731.2.9 getMarshaledProperties	3373
6.731.2.10 getMaxInactivityDuration	3373
6.731.2.11 getMaxInactivityDurationInitialDelay	3374
6.731.2.12 getProperties	3374
6.731.2.13 getProperties	3374
6.731.2.14 getVersion	3374
6.731.2.15 isCacheEnabled	3374
6.731.2.16 isMarshalAware	3374
6.731.2.17 isSizePrefixDisabled	3375
6.731.2.18 isStackTraceEnabled	3375
6.731.2.19 isTcpNoDelayEnabled	3375
6.731.2.20 isTightEncodingEnabled	3375
6.731.2.21 isValid	3375
6.731.2.22 isWireFormatInfo	3376
6.731.2.23 setCacheEnabled	3376
6.731.2.24 setCacheSize	3376
6.731.2.25 setMagic	3376
6.731.2.26 setMarshaledProperties	3376
6.731.2.27 setMaxInactivityDuration	3376
6.731.2.28 setMaxInactivityDurationInitialDelay	3377
6.731.2.29 setProperties	3377
6.731.2.30 setSizePrefixDisabled	3377
6.731.2.31 setStackTraceEnabled	3377
6.731.2.32 setTcpNoDelayEnabled	3377
6.731.2.33 setTightEncodingEnabled	3377
6.731.2.34 setVersion	3378
6.731.2.35 toString	3378
6.731.2.36 visit	3378
6.731.3 Field Documentation	3378
6.731.3.1 ID_WIREFORMATINFO	3378
6.732 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller Class Reference	3379
6.732.1 Detailed Description	3379

6.732.2 Constructor & Destructor Documentation	3380
6.732.2.1 WireFormatInfoMarshaller	3380
6.732.2.2 ~WireFormatInfoMarshaller	3380
6.732.3 Member Function Documentation	3380
6.732.3.1 createObject	3380
6.732.3.2 getDataStructureType	3380
6.732.3.3 looseMarshal	3380
6.732.3.4 looseUnmarshal	3381
6.732.3.5 tightMarshal1	3381
6.732.3.6 tightMarshal2	3381
6.732.3.7 tightUnmarshal	3382
6.733activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference	3383
6.733.1 Detailed Description	3383
6.733.2 Constructor & Destructor Documentation	3384
6.733.2.1 WireFormatInfoMarshaller	3384
6.733.2.2 ~WireFormatInfoMarshaller	3384
6.733.3 Member Function Documentation	3384
6.733.3.1 createObject	3384
6.733.3.2 getDataStructureType	3384
6.733.3.3 looseMarshal	3384
6.733.3.4 looseUnmarshal	3385
6.733.3.5 tightMarshal1	3385
6.733.3.6 tightMarshal2	3385
6.733.3.7 tightUnmarshal	3386
6.734activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference	3387
6.734.1 Detailed Description	3387
6.734.2 Constructor & Destructor Documentation	3388
6.734.2.1 WireFormatInfoMarshaller	3388
6.734.2.2 ~WireFormatInfoMarshaller	3388
6.734.3 Member Function Documentation	3388
6.734.3.1 createObject	3388
6.734.3.2 getDataStructureType	3388
6.734.3.3 looseMarshal	3388
6.734.3.4 looseUnmarshal	3389
6.734.3.5 tightMarshal1	3389

6.734.3.6 tightMarshal2	3389
6.734.3.7 tightUnmarshal	3390
6.735activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference	3391
6.735.1 Detailed Description	3391
6.735.2 Constructor & Destructor Documentation	3392
6.735.2.1 WireFormatInfoMarshaller	3392
6.735.2.2 ~WireFormatInfoMarshaller	3392
6.735.3 Member Function Documentation	3392
6.735.3.1 createObject	3392
6.735.3.2 getDataStructureType	3392
6.735.3.3 looseMarshal	3392
6.735.3.4 looseUnmarshal	3393
6.735.3.5 tightMarshal1	3393
6.735.3.6 tightMarshal2	3393
6.735.3.7 tightUnmarshal	3394
6.736activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference	3395
6.736.1 Detailed Description	3395
6.736.2 Constructor & Destructor Documentation	3396
6.736.2.1 WireFormatInfoMarshaller	3396
6.736.2.2 ~WireFormatInfoMarshaller	3396
6.736.3 Member Function Documentation	3396
6.736.3.1 createObject	3396
6.736.3.2 getDataStructureType	3396
6.736.3.3 looseMarshal	3396
6.736.3.4 looseUnmarshal	3397
6.736.3.5 tightMarshal1	3397
6.736.3.6 tightMarshal2	3397
6.736.3.7 tightUnmarshal	3398
6.737activemq::wireformat::WireFormatNegotiator Class Reference	3399
6.737.1 Detailed Description	3399
6.737.2 Constructor & Destructor Documentation	3399
6.737.2.1 WireFormatNegotiator	3399
6.737.2.2 ~WireFormatNegotiator	3399
6.738activemq::wireformat::WireFormatRegistry Class Reference	3400
6.738.1 Detailed Description	3400

6.738.2 Constructor & Destructor Documentation	3401
6.738.2.1 ~WireFormatRegistry	3401
6.738.3 Member Function Documentation	3401
6.738.3.1 findFactory	3401
6.738.3.2 getInstance	3401
6.738.3.3 getWireFormatNames	3401
6.738.3.4 registerFactory	3402
6.738.3.5 unregisterFactory	3402
6.739activemq::transport::inactivity::WriteChecker Class Reference	3403
6.739.1 Detailed Description	3403
6.739.2 Constructor & Destructor Documentation	3403
6.739.2.1 WriteChecker	3403
6.739.2.2 ~WriteChecker	3403
6.739.3 Member Function Documentation	3403
6.739.3.1 run	3403
6.740decaf::io::Writer Class Reference	3404
6.740.1 Constructor & Destructor Documentation	3404
6.740.1.1 ~Writer	3404
6.740.2 Member Function Documentation	3404
6.740.2.1 getOutputStream	3404
6.740.2.2 setOutputStream	3404
6.740.2.3 write	3404
6.740.2.4 writeByte	3405
6.741decaf::security::auth::x500::X500Principal Class Reference	3406
6.741.1 Constructor & Destructor Documentation	3406
6.741.1.1 ~X500Principal	3406
6.741.2 Member Function Documentation	3406
6.741.2.1 getEncoded	3406
6.741.2.2 getName	3406
6.741.2.3 hashCode	3406
6.742decaf::security::cert::X509Certificate Class Reference	3407
6.742.1 Detailed Description	3407
6.742.2 Constructor & Destructor Documentation	3407
6.742.2.1 ~X509Certificate	3407
6.742.3 Member Function Documentation	3407
6.742.3.1 checkValidity	3407

6.742.3.2	checkValidity	3408
6.742.3.3	getBasicConstraints	3408
6.742.3.4	getIssuerUniqueID	3408
6.742.3.5	getIssuerX500Principal	3408
6.742.3.6	getKeyUsage	3408
6.742.3.7	getNotAfter	3408
6.742.3.8	getNotBefore	3408
6.742.3.9	getSigAlgName	3408
6.742.3.10	getSigAlgOID	3409
6.742.3.11	getSigAlgParams	3409
6.742.3.12	getSignature	3409
6.742.3.13	getSubjectUniqueID	3409
6.742.3.14	getSubjectX500Principal	3409
6.742.3.15	getTBSCertificate	3409
6.742.3.16	getVersion	3409
6.743	activemq::commands::XATransactionId Class Reference	3410
6.743.1	Member Typedef Documentation	3411
6.743.1.1	COMPARATOR	3411
6.743.2	Constructor & Destructor Documentation	3411
6.743.2.1	XATransactionId	3411
6.743.2.2	XATransactionId	3411
6.743.2.3	~XATransactionId	3411
6.743.3	Member Function Documentation	3411
6.743.3.1	cloneDataStructure	3411
6.743.3.2	compareTo	3411
6.743.3.3	copyDataStructure	3411
6.743.3.4	equals	3412
6.743.3.5	equals	3412
6.743.3.6	getBranchQualifier	3412
6.743.3.7	getBranchQualifier	3412
6.743.3.8	getDataStructureType	3412
6.743.3.9	getFormatId	3413
6.743.3.10	getGlobalTransactionId	3413
6.743.3.11	getGlobalTransactionId	3413
6.743.3.12	operator<	3413
6.743.3.13	operator=	3413

6.743.3.14	operator==	3413
6.743.3.15	setBranchQualifier	3413
6.743.3.16	setFormatId	3413
6.743.3.17	setGlobalTransactionId	3413
6.743.3.18	toString	3413
6.743.4	Field Documentation	3414
6.743.4.1	branchQualifier	3414
6.743.4.2	formatId	3414
6.743.4.3	globalTransactionId	3414
6.743.4.4	ID_XATRANSACTIONID	3414
6.744	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class	
	Reference	3415
6.744.1	Detailed Description	3415
6.744.2	Constructor & Destructor Documentation	3416
6.744.2.1	XATransactionIdMarshaller	3416
6.744.2.2	~XATransactionIdMarshaller	3416
6.744.3	Member Function Documentation	3416
6.744.3.1	createObject	3416
6.744.3.2	getDataStructureType	3416
6.744.3.3	looseMarshal	3416
6.744.3.4	looseUnmarshal	3417
6.744.3.5	tightMarshal1	3417
6.744.3.6	tightMarshal2	3417
6.744.3.7	tightUnmarshal	3418
6.745	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller Class	
	Reference	3419
6.745.1	Detailed Description	3419
6.745.2	Constructor & Destructor Documentation	3420
6.745.2.1	XATransactionIdMarshaller	3420
6.745.2.2	~XATransactionIdMarshaller	3420
6.745.3	Member Function Documentation	3420
6.745.3.1	createObject	3420
6.745.3.2	getDataStructureType	3420
6.745.3.3	looseMarshal	3420
6.745.3.4	looseUnmarshal	3421
6.745.3.5	tightMarshal1	3421
6.745.3.6	tightMarshal2	3421

6.745.3.7 tightUnmarshal	3422
6.746activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller Class	
Reference	3423
6.746.1 Detailed Description	3423
6.746.2 Constructor & Destructor Documentation	3424
6.746.2.1 XATransactionIdMarshaller	3424
6.746.2.2 ~XATransactionIdMarshaller	3424
6.746.3 Member Function Documentation	3424
6.746.3.1 createObject	3424
6.746.3.2 getDataStructureType	3424
6.746.3.3 looseMarshal	3424
6.746.3.4 looseUnmarshal	3425
6.746.3.5 tightMarshal1	3425
6.746.3.6 tightMarshal2	3425
6.746.3.7 tightUnmarshal	3426
6.747activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class	
Reference	3427
6.747.1 Detailed Description	3427
6.747.2 Constructor & Destructor Documentation	3428
6.747.2.1 XATransactionIdMarshaller	3428
6.747.2.2 ~XATransactionIdMarshaller	3428
6.747.3 Member Function Documentation	3428
6.747.3.1 createObject	3428
6.747.3.2 getDataStructureType	3428
6.747.3.3 looseMarshal	3428
6.747.3.4 looseUnmarshal	3429
6.747.3.5 tightMarshal1	3429
6.747.3.6 tightMarshal2	3429
6.747.3.7 tightUnmarshal	3430
6.748activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller Class	
Reference	3431
6.748.1 Detailed Description	3431
6.748.2 Constructor & Destructor Documentation	3432
6.748.2.1 XATransactionIdMarshaller	3432
6.748.2.2 ~XATransactionIdMarshaller	3432
6.748.3 Member Function Documentation	3432
6.748.3.1 createObject	3432

6.748.3.2	getDataStructureType	3432
6.748.3.3	looseMarshal	3432
6.748.3.4	looseUnmarshal	3433
6.748.3.5	tightMarshal1	3433
6.748.3.6	tightMarshal2	3433
6.748.3.7	tightUnmarshal	3434
7	File Documentation	3435
7.1	src/main/activemq/cmsutil/CachedConsumer.h File Reference	3435
7.2	src/main/activemq/cmsutil/CachedProducer.h File Reference	3436
7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference	3437
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference	3438
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference	3439
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference	3440
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference	3441
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference	3442
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference	3443
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference	3444
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference	3445
7.12	src/main/activemq/cmsutil/SessionCallback.h File Reference	3446
7.13	src/main/activemq/cmsutil/SessionPool.h File Reference	3447
7.14	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference	3448
7.15	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference	3449
7.16	src/main/activemq/commands/ActiveMQDestination.h File Reference	3450
7.17	src/main/activemq/commands/ActiveMQMapMessage.h File Reference	3451
7.18	src/main/activemq/commands/ActiveMQMessage.h File Reference	3452
7.19	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference	3453
7.20	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference	3454
7.21	src/main/activemq/commands/ActiveMQQueue.h File Reference	3455
7.22	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference	3456
7.23	src/main/activemq/commands/ActiveMQTempDestination.h File Reference	3457
7.24	src/main/activemq/commands/ActiveMQTempQueue.h File Reference	3458
7.25	src/main/activemq/commands/ActiveMQTempTopic.h File Reference	3459
7.26	src/main/activemq/commands/ActiveMQTextMessage.h File Reference	3460
7.27	src/main/activemq/commands/ActiveMQTopic.h File Reference	3461
7.28	src/main/activemq/commands/BaseCommand.h File Reference	3462
7.29	src/main/activemq/commands/BaseDataStructure.h File Reference	3463

7.30	src/main/activemq/commands/BooleanExpression.h File Reference	3464
7.31	src/main/activemq/commands/BrokerError.h File Reference	3465
7.32	src/main/activemq/commands/BrokerId.h File Reference	3466
7.33	src/main/activemq/commands/BrokerInfo.h File Reference	3467
7.34	src/main/activemq/commands/Command.h File Reference	3468
7.35	src/main/activemq/commands/ConnectionControl.h File Reference	3469
7.36	src/main/activemq/commands/ConnectionError.h File Reference	3470
7.37	src/main/activemq/commands/ConnectionId.h File Reference	3471
7.38	src/main/activemq/commands/ConnectionInfo.h File Reference	3472
7.39	src/main/activemq/commands/ConsumerControl.h File Reference	3473
7.40	src/main/activemq/commands/ConsumerId.h File Reference	3474
7.41	src/main/activemq/commands/ConsumerInfo.h File Reference	3475
7.42	src/main/activemq/commands/ControlCommand.h File Reference	3476
7.43	src/main/activemq/commands/DataArrayResponse.h File Reference	3477
7.44	src/main/activemq/commands/DataResponse.h File Reference	3478
7.45	src/main/activemq/commands/DataStructure.h File Reference	3479
7.46	src/main/activemq/commands/DestinationInfo.h File Reference	3480
7.47	src/main/activemq/commands/DiscoveryEvent.h File Reference	3481
7.48	src/main/activemq/commands/ExceptionResponse.h File Reference	3482
7.49	src/main/activemq/commands/FlushCommand.h File Reference	3483
7.50	src/main/activemq/commands/IntegerResponse.h File Reference	3484
7.51	src/main/activemq/commands/JournalQueueAck.h File Reference	3485
7.52	src/main/activemq/commands/JournalTopicAck.h File Reference	3486
7.53	src/main/activemq/commands/JournalTrace.h File Reference	3487
7.54	src/main/activemq/commands/JournalTransaction.h File Reference	3488
7.55	src/main/activemq/commands/KeepAliveInfo.h File Reference	3489
7.56	src/main/activemq/commands/LastPartialCommand.h File Reference	3490
7.57	src/main/activemq/commands/LocalTransactionId.h File Reference	3491
7.58	src/main/activemq/commands/Message.h File Reference	3492
7.59	src/main/cms/Message.h File Reference	3493
7.60	src/main/activemq/commands/MessageAck.h File Reference	3494
7.61	src/main/activemq/commands/MessageDispatch.h File Reference	3495
7.62	src/main/activemq/commands/MessageDispatchNotification.h File Reference . . .	3496
7.63	src/main/activemq/commands/MessageId.h File Reference	3497
7.64	src/main/activemq/commands/MessagePull.h File Reference	3498
7.65	src/main/activemq/commands/NetworkBridgeFilter.h File Reference	3499

7.66	src/main/activemq/commands/PartialCommand.h File Reference	3500
7.67	src/main/activemq/commands/ProducerAck.h File Reference	3501
7.68	src/main/activemq/commands/ProducerId.h File Reference	3502
7.69	src/main/activemq/commands/ProducerInfo.h File Reference	3503
7.70	src/main/activemq/commands/RemoveInfo.h File Reference	3504
7.71	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference	3505
7.72	src/main/activemq/commands/ReplayCommand.h File Reference	3506
7.73	src/main/activemq/commands/Response.h File Reference	3507
7.74	src/main/activemq/commands/SessionId.h File Reference	3508
7.75	src/main/activemq/commands/SessionInfo.h File Reference	3509
7.76	src/main/activemq/commands/ShutdownInfo.h File Reference	3510
7.77	src/main/activemq/commands/SubscriptionInfo.h File Reference	3511
7.78	src/main/activemq/commands/TransactionId.h File Reference	3512
7.79	src/main/activemq/commands/TransactionInfo.h File Reference	3513
7.80	src/main/activemq/commands/WireFormatInfo.h File Reference	3514
7.81	src/main/activemq/commands/XATransactionId.h File Reference	3515
7.82	src/main/activemq/core/ActiveMQAckHandler.h File Reference	3516
7.83	src/main/activemq/core/ActiveMQConnection.h File Reference	3517
7.84	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference	3518
7.85	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference	3519
7.86	src/main/activemq/core/ActiveMQConnectionSupport.h File Reference	3520
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference	3521
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference	3522
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference	3523
7.90	src/main/activemq/core/ActiveMQSession.h File Reference	3524
7.91	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference	3525
7.92	src/main/activemq/core/ActiveMQTransactionContext.h File Reference	3526
7.93	src/main/activemq/core/DispatchData.h File Reference	3527
7.94	src/main/activemq/core/Dispatcher.h File Reference	3528
7.95	src/main/activemq/core/MessageDispatchChannel.h File Reference	3529
7.96	src/main/activemq/core/Synchronization.h File Reference	3530
7.97	src/main/activemq/exceptions/ActiveMQException.h File Reference	3531
7.98	src/main/activemq/exceptions/BrokerException.h File Reference	3532
7.99	src/main/activemq/exceptions/ExceptionDefines.h File Reference	3533
7.99.1	Define Documentation	3533
7.99.1.1	AMQ_CATCH_EXCEPTION_CONVERT	3533

7.99.1.2	AMQ_CATCH_NOTHROW	3533
7.99.1.3	AMQ_CATCH_RETHROW	3534
7.99.1.4	AMQ_CATCHALL_NOTHROW	3534
7.99.1.5	AMQ_CATCHALL_THROW	3534
7.100	src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference	3536
7.100.1	Define Documentation	3536
7.100.1.1	DECAF_CATCH_EXCEPTION_CONVERT	3536
7.100.1.2	DECAF_CATCH_NOTHROW	3536
7.100.1.3	DECAF_CATCH_RETHROW	3537
7.100.1.4	DECAF_CATCHALL_NOTHROW	3537
7.100.1.5	DECAF_CATCHALL_THROW	3537
7.101	src/main/activemq/io/LoggingInputStream.h File Reference	3539
7.102	src/main/activemq/io/LoggingOutputStream.h File Reference	3540
7.103	src/main/activemq/library/ActiveMQCPP.h File Reference	3541
7.104	src/main/activemq/state/CommandVisitor.h File Reference	3542
7.105	src/main/activemq/state/CommandVisitorAdapter.h File Reference	3543
7.106	src/main/activemq/state/ConnectionState.h File Reference	3545
7.107	src/main/activemq/state/ConnectionStateTracker.h File Reference	3546
7.108	src/main/activemq/state/ConsumerState.h File Reference	3547
7.109	src/main/activemq/state/ProducerState.h File Reference	3548
7.110	src/main/activemq/state/SessionState.h File Reference	3549
7.111	src/main/activemq/state/Tracked.h File Reference	3550
7.112	src/main/activemq/state/TransactionState.h File Reference	3551
7.113	src/main/activemq/threads/CompositeTask.h File Reference	3552
7.114	src/main/activemq/threads/CompositeTaskRunner.h File Reference	3553
7.115	src/main/activemq/threads/DedicatedTaskRunner.h File Reference	3554
7.116	src/main/activemq/threads/Task.h File Reference	3555
7.117	src/main/activemq/threads/TaskRunner.h File Reference	3556
7.118	src/main/activemq/transport/AbstractTransportFactory.h File Reference	3557
7.119	src/main/activemq/transport/CompositeTransport.h File Reference	3558
7.120	src/main/activemq/transport/correlator/FutureResponse.h File Reference	3559
7.121	src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference	3560
7.122	src/main/activemq/transport/DefaultTransportListener.h File Reference	3561
7.123	src/main/activemq/transport/failover/BackupTransport.h File Reference	3562
7.124	src/main/activemq/transport/failover/BackupTransportPool.h File Reference	3563
7.125	src/main/activemq/transport/failover/CloseTransportsTask.h File Reference	3564

7.126src/main/activemq/transport/failover/FailoverTransport.h File Reference	3565
7.127src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference	3566
7.128src/main/activemq/transport/failover/FailoverTransportListener.h File Reference	3567
7.129src/main/activemq/transport/failover/URIPool.h File Reference	3568
7.130src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference	3569
7.131src/main/activemq/transport/inactivity/ReadChecker.h File Reference	3570
7.132src/main/activemq/transport/inactivity/WriteChecker.h File Reference	3571
7.133src/main/activemq/transport/IOTransport.h File Reference	3572
7.134src/main/activemq/transport/logging/LoggingTransport.h File Reference	3573
7.135src/main/activemq/transport/mock/InternalCommandListener.h File Reference	3574
7.136src/main/activemq/transport/mock/MockTransport.h File Reference	3575
7.137src/main/activemq/transport/mock/MockTransportFactory.h File Reference	3576
7.138src/main/activemq/transport/mock/ResponseBuilder.h File Reference	3577
7.139src/main/activemq/transport/tcp/TcpTransport.h File Reference	3578
7.140src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference	3579
7.141src/main/activemq/transport/Transport.h File Reference	3580
7.142src/main/activemq/transport/TransportFactory.h File Reference	3581
7.143src/main/activemq/transport/TransportFilter.h File Reference	3582
7.144src/main/activemq/transport/TransportListener.h File Reference	3583
7.145src/main/activemq/transport/TransportRegistry.h File Reference	3584
7.146src/main/activemq/util/ActiveMQProperties.h File Reference	3585
7.147src/main/activemq/util/CMSExceptionSupport.h File Reference	3586
7.147.1 Define Documentation	3586
7.147.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION	3586
7.148src/main/activemq/util/CompositeData.h File Reference	3588
7.149src/main/activemq/util/Config.h File Reference	3589
7.149.1 Define Documentation	3589
7.149.1.1 AMQCPP_API	3589
7.149.1.2 HAVE_PTHREAD_H	3589
7.149.1.3 HAVE_UUID_T	3589
7.149.1.4 HAVE_UUID_UUID_H	3589
7.150src/main/cms/Config.h File Reference	3590
7.150.1 Define Documentation	3590
7.150.1.1 CMS_API	3590
7.151src/main/decaf/util/Config.h File Reference	3591
7.151.1 Define Documentation	3591

7.151.1.1 DECAF_ API	3591
7.151.1.2 DECAF_ UNUSED	3591
7.152src/main/activemq/util/LongSequenceGenerator.h File Reference	3592
7.153src/main/activemq/util/MemoryUsage.h File Reference	3593
7.154src/main/activemq/util/PrimitiveList.h File Reference	3594
7.155src/main/activemq/util/PrimitiveMap.h File Reference	3595
7.156src/main/activemq/util/PrimitiveValueConverter.h File Reference	3596
7.157src/main/activemq/util/PrimitiveValueNode.h File Reference	3597
7.158src/main/activemq/util/URISupport.h File Reference	3598
7.159src/main/activemq/util/Usage.h File Reference	3599
7.160src/main/activemq/wireformat/MarshalAware.h File Reference	3600
7.161src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference	3601
7.162src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference	3602
7.163src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference	3603
7.164src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference	3604
7.165src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference	3605
7.166src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference	3606
7.167src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h File Reference	3607
7.168src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference	3608
7.169src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference	3609
7.170src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference	3610
7.171src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference	3611
7.172src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference	3612
7.173src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference	3613
7.174src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference	3614
7.175src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h File Reference	3615

7.176	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
	File Reference	3616
7.177	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	
	File Reference	3617
7.178	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h	
	File Reference	3618
7.179	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h	
	File Reference	3619
7.180	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
	File Reference	3620
7.181	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
	File Reference	3621
7.182	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h	
	File Reference	3622
7.183	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h	
	File Reference	3623
7.184	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h	
	File Reference	3624
7.185	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
	File Reference	3625
7.186	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
	File Reference	3626
7.187	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h	
	File Reference	3627
7.188	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h	
	File Reference	3628
7.189	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h	
	File Reference	3629
7.190	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
	File Reference	3630
7.191	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h	
	File Reference	3631
7.192	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h	
	File Reference	3632
7.193	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h	
	File Reference	3633
7.194	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h	
	File Reference	3634
7.195	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h	
	File Reference	3635
7.196	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h	
	File Reference	3636
7.197	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h	
	File Reference	3637

7.198	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h	
	File Reference	3638
7.199	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h	
	File Reference	3639
7.200	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h	
	File Reference	3640
7.201	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h	
	File Reference	3641
7.202	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h	
	File Reference	3642
7.203	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h	
	File Reference	3643
7.204	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h	
	File Reference	3644
7.205	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h	
	File Reference	3645
7.206	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h	
	File Reference	3646
7.207	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h	
	File Reference	3647
7.208	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h	
	File Reference	3648
7.209	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h	
	File Reference	3649
7.210	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h	
	File Reference	3650
7.211	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h	
	File Reference	3651
7.212	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h	
	File Reference	3652
7.213	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h	
	File Reference	3653
7.214	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h	
	File Reference	3654
7.215	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h	
	File Reference	3655
7.216	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h	
	File Reference	3656
7.217	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h	
	File Reference	3657
7.218	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h	
	File Reference	3658
7.219	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h	
	File Reference	3659

7.220	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h	
	File Reference	3660
7.221	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h	
	File Reference	3661
7.222	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h	
	File Reference	3662
7.223	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h	
	File Reference	3663
7.224	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	
	File Reference	3664
7.225	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	
	File Reference	3665
7.226	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	
	File Reference	3666
7.227	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	
	File Reference	3667
7.228	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	
	File Reference	3668
7.229	src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	
	File Reference	3669
7.230	src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	
	File Reference	3670
7.231	src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	
	File Reference	3671
7.232	src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	
	File Reference	3672
7.233	src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	
	File Reference	3673
7.234	src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h	File
	Reference	3674
7.235	src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	File
	Reference	3675
7.236	src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	File
	Reference	3676
7.237	src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	File
	Reference	3677
7.238	src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h	File
	Reference	3678
7.239	src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h	File
	Reference	3679
7.240	src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h	File
	Reference	3680
7.241	src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h	File
	Reference	3681

7.242src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h File	
Reference	3682
7.243src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h File	
Reference	3683
7.244src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
File Reference	3684
7.245src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
File Reference	3685
7.246src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
File Reference	3686
7.247src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
File Reference	3687
7.248src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
File Reference	3688
7.249src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
File Reference	3689
7.250src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
File Reference	3690
7.251src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
File Reference	3691
7.252src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
File Reference	3692
7.253src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
File Reference	3693
7.254src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h	
File Reference	3694
7.255src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h	
File Reference	3695
7.256src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	
File Reference	3696
7.257src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	
File Reference	3697
7.258src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h	
File Reference	3698
7.259src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h	
File Reference	3699
7.260src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h	
File Reference	3700
7.261src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h	
File Reference	3701
7.262src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h	
File Reference	3702
7.263src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h	
File Reference	3703

7.264	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h	
	File Reference	3704
7.265	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h	
	File Reference	3705
7.266	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h	
	File Reference	3706
7.267	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h	
	File Reference	3707
7.268	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h	
	File Reference	3708
7.269	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h	
	File Reference	3709
7.270	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h	
	File Reference	3710
7.271	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	
	File Reference	3711
7.272	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	
	File Reference	3712
7.273	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h	
	File Reference	3713
7.274	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h	
	File Reference	3714
7.275	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h	
	File Reference	3715
7.276	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	
	File Reference	3716
7.277	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	
	File Reference	3717
7.278	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h	
	File Reference	3718
7.279	src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
	File Reference	3719
7.280	src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h	
	File Reference	3720
7.281	src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
	File Reference	3721
7.282	src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
	File Reference	3722
7.283	src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
	File Reference	3723
7.284	src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
	File Reference	3724
7.285	src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h	
	File Reference	3725

7.286	src/main/activemq/wireformat/openwire/marshall/v3/DataArrayResponseMarshaller.h	
	File Reference	3726
7.287	src/main/activemq/wireformat/openwire/marshall/v4/DataArrayResponseMarshaller.h	
	File Reference	3727
7.288	src/main/activemq/wireformat/openwire/marshall/v5/DataArrayResponseMarshaller.h	
	File Reference	3728
7.289	src/main/activemq/wireformat/openwire/marshall/v1/DataResponseMarshaller.h	
	File Reference	3729
7.290	src/main/activemq/wireformat/openwire/marshall/v2/DataResponseMarshaller.h	
	File Reference	3730
7.291	src/main/activemq/wireformat/openwire/marshall/v3/DataResponseMarshaller.h	
	File Reference	3731
7.292	src/main/activemq/wireformat/openwire/marshall/v4/DataResponseMarshaller.h	
	File Reference	3732
7.293	src/main/activemq/wireformat/openwire/marshall/v5/DataResponseMarshaller.h	
	File Reference	3733
7.294	src/main/activemq/wireformat/openwire/marshall/v1/DestinationInfoMarshaller.h	
	File Reference	3734
7.295	src/main/activemq/wireformat/openwire/marshall/v2/DestinationInfoMarshaller.h	
	File Reference	3735
7.296	src/main/activemq/wireformat/openwire/marshall/v3/DestinationInfoMarshaller.h	
	File Reference	3736
7.297	src/main/activemq/wireformat/openwire/marshall/v4/DestinationInfoMarshaller.h	
	File Reference	3737
7.298	src/main/activemq/wireformat/openwire/marshall/v5/DestinationInfoMarshaller.h	
	File Reference	3738
7.299	src/main/activemq/wireformat/openwire/marshall/v1/DiscoveryEventMarshaller.h	
	File Reference	3739
7.300	src/main/activemq/wireformat/openwire/marshall/v2/DiscoveryEventMarshaller.h	
	File Reference	3740
7.301	src/main/activemq/wireformat/openwire/marshall/v3/DiscoveryEventMarshaller.h	
	File Reference	3741
7.302	src/main/activemq/wireformat/openwire/marshall/v4/DiscoveryEventMarshaller.h	
	File Reference	3742
7.303	src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEventMarshaller.h	
	File Reference	3743
7.304	src/main/activemq/wireformat/openwire/marshall/v1/ExceptionResponseMarshaller.h	
	File Reference	3744
7.305	src/main/activemq/wireformat/openwire/marshall/v2/ExceptionResponseMarshaller.h	
	File Reference	3745
7.306	src/main/activemq/wireformat/openwire/marshall/v3/ExceptionResponseMarshaller.h	
	File Reference	3746
7.307	src/main/activemq/wireformat/openwire/marshall/v4/ExceptionResponseMarshaller.h	
	File Reference	3747

7.308	src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h	
	File Reference	3748
7.309	src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h	
	File Reference	3749
7.310	src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h	
	File Reference	3750
7.311	src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	
	File Reference	3751
7.312	src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	
	File Reference	3752
7.313	src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	
	File Reference	3753
7.314	src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
	File Reference	3754
7.315	src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h	
	File Reference	3755
7.316	src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
	File Reference	3756
7.317	src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
	File Reference	3757
7.318	src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
	File Reference	3758
7.319	src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
	File Reference	3759
7.320	src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h	
	File Reference	3760
7.321	src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
	File Reference	3761
7.322	src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
	File Reference	3762
7.323	src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
	File Reference	3763
7.324	src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
	File Reference	3764
7.325	src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h	
	File Reference	3765
7.326	src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
	File Reference	3766
7.327	src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
	File Reference	3767
7.328	src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
	File Reference	3768
7.329	src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h	
	File Reference	3769

7.330src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h	
File Reference	3770
7.331src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	
File Reference	3771
7.332src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
File Reference	3772
7.333src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h	
File Reference	3773
7.334src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
File Reference	3774
7.335src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h	
File Reference	3775
7.336src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	
File Reference	3776
7.337src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
File Reference	3777
7.338src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
File Reference	3778
7.339src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h	
File Reference	3779
7.340src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h	
File Reference	3780
7.341src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	
File Reference	3781
7.342src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	
File Reference	3782
7.343src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h	
File Reference	3783
7.344src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
File Reference	3784
7.345src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h	
File Reference	3785
7.346src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	
File Reference	3786
7.347src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
File Reference	3787
7.348src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
File Reference	3788
7.349src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
File Reference	3789
7.350src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h	
File Reference	3790
7.351src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	
File Reference	3791

7.352	src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
	File Reference	3792
7.353	src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h	
	File Reference	3793
7.354	src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h	File
	Reference	3794
7.355	src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h	File
	Reference	3795
7.356	src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h	File
	Reference	3796
7.357	src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	File
	Reference	3797
7.358	src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h	File
	Reference	3798
7.359	src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h	
	File Reference	3799
7.360	src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h	
	File Reference	3800
7.361	src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h	
	File Reference	3801
7.362	src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h	
	File Reference	3802
7.363	src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h	
	File Reference	3803
7.364	src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h	
	File Reference	3804
7.365	src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h	
	File Reference	3805
7.366	src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h	
	File Reference	3806
7.367	src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
	File Reference	3807
7.368	src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h	
	File Reference	3808
7.369	src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h	
	File Reference	3809
7.370	src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h	
	File Reference	3810
7.371	src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h	
	File Reference	3811
7.372	src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
	File Reference	3812
7.373	src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h	
	File Reference	3813

7.374	src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h	File	
	Reference		3814
7.375	src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h	File	
	Reference		3815
7.376	src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	File	
	Reference		3816
7.377	src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h	File	
	Reference		3817
7.378	src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h	File	
	Reference		3818
7.379	src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h	File	
	Reference		3819
7.380	src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h	File	
	Reference		3820
7.381	src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	File	
	Reference		3821
7.382	src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	File	
	Reference		3822
7.383	src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h	File	
	Reference		3823
7.384	src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h		
	File Reference		3824
7.385	src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h		
	File Reference		3825
7.386	src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h		
	File Reference		3826
7.387	src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h		
	File Reference		3827
7.388	src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h		
	File Reference		3828
7.389	src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h		
	File Reference		3829
7.390	src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h		
	File Reference		3830
7.391	src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h		
	File Reference		3831
7.392	src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h		
	File Reference		3832
7.393	src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h		
	File Reference		3833
7.394	src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h		
	File Reference		3834
7.395	src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h		
	File Reference		3835

7.396	src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	
	File Reference	3836
7.397	src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
	File Reference	3837
7.398	src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h	
	File Reference	3838
7.399	src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h	
	File Reference	3839
7.400	src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h	
	File Reference	3840
7.401	src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	
	File Reference	3841
7.402	src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h	
	File Reference	3842
7.403	src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h	
	File Reference	3843
7.404	src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h	
	File Reference	3844
7.405	src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h	
	File Reference	3845
7.406	src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	
	File Reference	3846
7.407	src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h	
	File Reference	3847
7.408	src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h	
	File Reference	3848
7.409	src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h	
	File Reference	3849
7.410	src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h	
	File Reference	3850
7.411	src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	
	File Reference	3851
7.412	src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h	
	File Reference	3852
7.413	src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h	
	File Reference	3853
7.414	src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h	
	File Reference	3854
7.415	src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h	
	File Reference	3855
7.416	src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	
	File Reference	3856
7.417	src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h	
	File Reference	3857

7.418	src/main/activemq/wireformat/openwire/marshall/v5/RemoveInfoMarshaller.h	
	File Reference	3858
7.419	src/main/activemq/wireformat/openwire/marshall/v1/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3859
7.420	src/main/activemq/wireformat/openwire/marshall/v2/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3860
7.421	src/main/activemq/wireformat/openwire/marshall/v3/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3861
7.422	src/main/activemq/wireformat/openwire/marshall/v4/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3862
7.423	src/main/activemq/wireformat/openwire/marshall/v5/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3863
7.424	src/main/activemq/wireformat/openwire/marshall/v1/ReplayCommandMarshaller.h	
	File Reference	3864
7.425	src/main/activemq/wireformat/openwire/marshall/v2/ReplayCommandMarshaller.h	
	File Reference	3865
7.426	src/main/activemq/wireformat/openwire/marshall/v3/ReplayCommandMarshaller.h	
	File Reference	3866
7.427	src/main/activemq/wireformat/openwire/marshall/v4/ReplayCommandMarshaller.h	
	File Reference	3867
7.428	src/main/activemq/wireformat/openwire/marshall/v5/ReplayCommandMarshaller.h	
	File Reference	3868
7.429	src/main/activemq/wireformat/openwire/marshall/v1/ResponseMarshaller.h	File
	Reference	3869
7.430	src/main/activemq/wireformat/openwire/marshall/v2/ResponseMarshaller.h	File
	Reference	3870
7.431	src/main/activemq/wireformat/openwire/marshall/v3/ResponseMarshaller.h	File
	Reference	3871
7.432	src/main/activemq/wireformat/openwire/marshall/v4/ResponseMarshaller.h	File
	Reference	3872
7.433	src/main/activemq/wireformat/openwire/marshall/v5/ResponseMarshaller.h	File
	Reference	3873
7.434	src/main/activemq/wireformat/openwire/marshall/v1/SessionIdMarshaller.h	File
	Reference	3874
7.435	src/main/activemq/wireformat/openwire/marshall/v2/SessionIdMarshaller.h	File
	Reference	3875
7.436	src/main/activemq/wireformat/openwire/marshall/v3/SessionIdMarshaller.h	File
	Reference	3876
7.437	src/main/activemq/wireformat/openwire/marshall/v4/SessionIdMarshaller.h	File
	Reference	3877
7.438	src/main/activemq/wireformat/openwire/marshall/v5/SessionIdMarshaller.h	File
	Reference	3878
7.439	src/main/activemq/wireformat/openwire/marshall/v1/SessionInfoMarshaller.h	
	File Reference	3879

7.440src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h	
File Reference	3880
7.441src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	
File Reference	3881
7.442src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h	
File Reference	3882
7.443src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h	
File Reference	3883
7.444src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h	
File Reference	3884
7.445src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h	
File Reference	3885
7.446src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	
File Reference	3886
7.447src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h	
File Reference	3887
7.448src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h	
File Reference	3888
7.449src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	
File Reference	3889
7.450src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h	
File Reference	3890
7.451src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	
File Reference	3891
7.452src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h	
File Reference	3892
7.453src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h	
File Reference	3893
7.454src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h	
File Reference	3894
7.455src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h	
File Reference	3895
7.456src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	
File Reference	3896
7.457src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h	
File Reference	3897
7.458src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h	
File Reference	3898
7.459src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	
File Reference	3899
7.460src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h	
File Reference	3900
7.461src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h	
File Reference	3901

7.462	src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h File Reference	3902
7.463	src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h File Reference	3903
7.464	src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h File Reference	3904
7.465	src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h File Reference	3905
7.466	src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference	3906
7.467	src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h File Reference	3907
7.468	src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h File Reference	3908
7.469	src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h File Reference	3909
7.470	src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h File Reference	3910
7.471	src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h File Reference	3911
7.472	src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File Reference	3912
7.473	src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h File Reference	3913
7.474	src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference	3914
7.475	src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference	3915
7.476	src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Ref- erence	3916
7.477	src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Refer- ence	3917
7.478	src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference . .	3918
7.479	src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference	3919
7.480	src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h File Reference	3920
7.481	src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h File Ref- erence	3921
7.482	src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference	3922
7.483	src/main/activemq/wireformat/stomp/StompFrame.h File Reference	3923
7.484	src/main/activemq/wireformat/stomp/StompHelper.h File Reference	3924
7.485	src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference	3925
7.486	src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference	3926
7.487	src/main/activemq/wireformat/WireFormat.h File Reference	3927

7.488src/main/activemq/wireformat/WireFormatFactory.h File Reference	3928
7.489src/main/activemq/wireformat/WireFormatNegotiator.h File Reference	3929
7.490src/main/activemq/wireformat/WireFormatRegistry.h File Reference	3930
7.491src/main/cms/BytesMessage.h File Reference	3931
7.492src/main/cms/Closeable.h File Reference	3932
7.493src/main/decaf/io/Closeable.h File Reference	3933
7.494src/main/cms/CMSException.h File Reference	3934
7.495src/main/cms/CMSProperties.h File Reference	3935
7.496src/main/cms/CMSSecurityException.h File Reference	3936
7.497src/main/cms/Connection.h File Reference	3937
7.498src/main/cms/ConnectionFactory.h File Reference	3938
7.499src/main/cms/ConnectionMetaData.h File Reference	3939
7.500src/main/cms/DeliveryMode.h File Reference	3940
7.501src/main/cms/Destination.h File Reference	3941
7.502src/main/cms/ExceptionListener.h File Reference	3942
7.503src/main/cms/IllegalStateException.h File Reference	3943
7.504src/main/decaf/lang/exceptions/IllegalStateException.h File Reference	3944
7.505src/main/cms/InvalidClientIdException.h File Reference	3945
7.506src/main/cms/InvalidDestinationException.h File Reference	3946
7.507src/main/cms/InvalidSelectorException.h File Reference	3947
7.508src/main/cms/MapMessage.h File Reference	3948
7.509src/main/cms/MessageConsumer.h File Reference	3949
7.510src/main/cms/MessageEOFException.h File Reference	3950
7.511src/main/cms/MessageFormatException.h File Reference	3951
7.512src/main/cms/MessageListener.h File Reference	3952
7.513src/main/cms/MessageNotReadableException.h File Reference	3953
7.514src/main/cms/MessageNotWriteableException.h File Reference	3954
7.515src/main/cms/MessageProducer.h File Reference	3955
7.516src/main/cms/ObjectMessage.h File Reference	3956
7.517src/main/cms/Queue.h File Reference	3957
7.518src/main/decaf/util/Queue.h File Reference	3958
7.519src/main/cms/QueueBrowser.h File Reference	3959
7.520src/main/cms/Session.h File Reference	3960
7.521src/main/cms/Startable.h File Reference	3961
7.522src/main/cms/Stoppable.h File Reference	3962
7.523src/main/cms/StreamMessage.h File Reference	3963

7.524src/main/cms/TemporaryQueue.h File Reference	3964
7.525src/main/cms/TemporaryTopic.h File Reference	3965
7.526src/main/cms/TextMessage.h File Reference	3966
7.527src/main/cms/Topic.h File Reference	3967
7.528src/main/cms/UnsupportedOperationException.h File Reference	3968
7.529src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference	3969
7.530src/main/decaf/internal/AprPool.h File Reference	3970
7.531src/main/decaf/internal/DecafRuntime.h File Reference	3971
7.532src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference	3972
7.533src/main/decaf/internal/io/StandardInputStream.h File Reference	3973
7.534src/main/decaf/internal/io/StandardOutputStream.h File Reference	3974
7.535src/main/decaf/internal/net/URIEncoderDecoder.h File Reference	3975
7.536src/main/decaf/internal/net/URIHelper.h File Reference	3976
7.537src/main/decaf/internal/net/URIType.h File Reference	3977
7.538src/main/decaf/internal/nio/BufferFactory.h File Reference	3978
7.539src/main/decaf/internal/nio/ByteBuffer.h File Reference	3979
7.540src/main/decaf/internal/nio/ByteBufferPerspective.h File Reference	3980
7.541src/main/decaf/internal/nio/CharArrayBuffer.h File Reference	3981
7.542src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference	3982
7.543src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference	3983
7.544src/main/decaf/internal/nio/IntArrayBuffer.h File Reference	3984
7.545src/main/decaf/internal/nio/LongArrayBuffer.h File Reference	3985
7.546src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference	3986
7.547src/main/decaf/internal/util/ByteArrayAdapter.h File Reference	3987
7.548src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference	3988
7.549src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference	3989
7.550src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference . . .	3990
7.551src/main/decaf/internal/util/concurrent/Transferer.h File Reference	3991
7.552src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference	3992
7.553src/main/decaf/internal/util/concurrent/TransferStack.h File Reference	3993
7.554src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference .	3994
7.555src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference	3995
7.556src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference . . .	3996
7.557src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference .	3997
7.558src/main/decaf/internal/util/HexStringParser.h File Reference	3998
7.559src/main/decaf/internal/util/TimerTaskHeap.h File Reference	3999

7.560src/main/decaf/io/BlockingByteArrayInputStream.h File Reference	4000
7.561src/main/decaf/io/BufferedInputStream.h File Reference	4001
7.562src/main/decaf/io/BufferedOutputStream.h File Reference	4002
7.563src/main/decaf/io/ByteArrayInputStream.h File Reference	4003
7.564src/main/decaf/io/ByteArrayOutputStream.h File Reference	4004
7.565src/main/decaf/io/DataInputStream.h File Reference	4005
7.566src/main/decaf/io/DataOutputStream.h File Reference	4006
7.567src/main/decaf/io/EOFException.h File Reference	4007
7.568src/main/decaf/io/FilterInputStream.h File Reference	4008
7.569src/main/decaf/io/FilterOutputStream.h File Reference	4009
7.570src/main/decaf/io/InputStream.h File Reference	4010
7.571src/main/decaf/io/InterruptedIOException.h File Reference	4011
7.572src/main/decaf/io/IOException.h File Reference	4012
7.573src/main/decaf/io/OutputStream.h File Reference	4013
7.574src/main/decaf/io/Reader.h File Reference	4014
7.575src/main/decaf/io/UnsupportedEncodingException.h File Reference	4015
7.576src/main/decaf/io/UTFDataFormatException.h File Reference	4016
7.577src/main/decaf/io/Writer.h File Reference	4017
7.578src/main/decaf/lang/Appendable.h File Reference	4018
7.579src/main/decaf/lang/Boolean.h File Reference	4019
7.580src/main/decaf/lang/Byte.h File Reference	4020
7.581src/main/decaf/lang/Character.h File Reference	4021
7.582src/main/decaf/lang/CharSequence.h File Reference	4022
7.583src/main/decaf/lang/Comparable.h File Reference	4023
7.584src/main/decaf/lang/Double.h File Reference	4024
7.585src/main/decaf/lang/Exception.h File Reference	4025
7.586src/main/decaf/lang/exceptions/ClassCastException.h File Reference	4026
7.587src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference	4027
7.588src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference . .	4028
7.589src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference . .	4029
7.590src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference . .	4030
7.591src/main/decaf/lang/exceptions/InterruptedException.h File Reference	4031
7.592src/main/decaf/lang/exceptions/InvalidStateException.h File Reference	4032
7.593src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference	4033
7.594src/main/decaf/lang/exceptions/NullPointerException.h File Reference	4034
7.595src/main/decaf/lang/exceptions/NumberFormatException.h File Reference	4035

7.596	src/main/decaf/lang/exceptions/RuntimeException.h File Reference	4036
7.597	src/main/decaf/lang/Float.h File Reference	4037
7.598	src/main/decaf/lang/Integer.h File Reference	4038
7.599	src/main/decaf/lang/Iterable.h File Reference	4039
7.600	src/main/decaf/lang/Long.h File Reference	4040
7.601	src/main/decaf/lang/Math.h File Reference	4041
7.602	src/main/decaf/lang/Number.h File Reference	4042
7.603	src/main/decaf/lang/Pointer.h File Reference	4043
7.604	src/main/decaf/lang/Runnable.h File Reference	4045
7.605	src/main/decaf/lang/Runtime.h File Reference	4046
7.606	src/main/decaf/lang/Short.h File Reference	4047
7.607	src/main/decaf/lang/System.h File Reference	4048
7.608	src/main/decaf/lang/Thread.h File Reference	4049
7.609	src/main/decaf/lang/ThreadGroup.h File Reference	4050
7.610	src/main/decaf/lang/Throwable.h File Reference	4051
7.611	src/main/decaf/net/BindException.h File Reference	4052
7.612	src/main/decaf/net/BufferedSocket.h File Reference	4053
7.613	src/main/decaf/net/ConnectException.h File Reference	4054
7.614	src/main/decaf/net/HttpRetryException.h File Reference	4055
7.615	src/main/decaf/net/MalformedURLException.h File Reference	4056
7.616	src/main/decaf/net/NoRouteToHostException.h File Reference	4057
7.617	src/main/decaf/net/PortUnreachableException.h File Reference	4058
7.618	src/main/decaf/net/ProtocolException.h File Reference	4059
7.619	src/main/decaf/net/ServerSocket.h File Reference	4060
7.620	src/main/decaf/net/Socket.h File Reference	4061
7.621	src/main/decaf/net/SocketError.h File Reference	4062
7.622	src/main/decaf/net/SocketException.h File Reference	4063
7.623	src/main/decaf/net/SocketFactory.h File Reference	4064
7.624	src/main/decaf/net/SocketInputStream.h File Reference	4065
7.625	src/main/decaf/net/SocketOutputStream.h File Reference	4066
7.626	src/main/decaf/net/SocketTimeoutException.h File Reference	4067
7.627	src/main/decaf/net/TcpSocket.h File Reference	4068
7.628	src/main/decaf/net/UnknownHostException.h File Reference	4069
7.629	src/main/decaf/net/UnknownServiceException.h File Reference	4070
7.630	src/main/decaf/net/URI.h File Reference	4071
7.631	src/main/decaf/net/URISyntaxException.h File Reference	4072

7.632src/main/decaf/net/URL.h File Reference	4073
7.633src/main/decaf/net/URLDecoder.h File Reference	4074
7.634src/main/decaf/net/URLEncoder.h File Reference	4075
7.635src/main/decaf/nio/Buffer.h File Reference	4076
7.636src/main/decaf/nio/BufferOverflowException.h File Reference	4077
7.637src/main/decaf/nio/BufferUnderflowException.h File Reference	4078
7.638src/main/decaf/nio/ByteBuffer.h File Reference	4079
7.639src/main/decaf/nio/CharBuffer.h File Reference	4080
7.640src/main/decaf/nio/DoubleBuffer.h File Reference	4081
7.641src/main/decaf/nio/FloatBuffer.h File Reference	4082
7.642src/main/decaf/nio/IntBuffer.h File Reference	4083
7.643src/main/decaf/nio/InvalidMarkException.h File Reference	4084
7.644src/main/decaf/nio/LongBuffer.h File Reference	4085
7.645src/main/decaf/nio/ReadOnlyBufferException.h File Reference	4086
7.646src/main/decaf/nio/ShortBuffer.h File Reference	4087
7.647src/main/decaf/security/auth/x500/X500Principal.h File Reference	4088
7.648src/main/decaf/security/cert/Certificate.h File Reference	4089
7.649src/main/decaf/security/cert/CertificateEncodingException.h File Reference . . .	4090
7.650src/main/decaf/security/cert/CertificateException.h File Reference	4091
7.651src/main/decaf/security/cert/CertificateExpiredException.h File Reference	4092
7.652src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference . .	4093
7.653src/main/decaf/security/cert/CertificateParsingException.h File Reference	4094
7.654src/main/decaf/security/cert/X509Certificate.h File Reference	4095
7.655src/main/decaf/security/GeneralSecurityException.h File Reference	4096
7.656src/main/decaf/security/InvalidKeyException.h File Reference	4097
7.657src/main/decaf/security/Key.h File Reference	4098
7.658src/main/decaf/security/KeyException.h File Reference	4099
7.659src/main/decaf/security/NoSuchAlgorithmException.h File Reference	4100
7.660src/main/decaf/security/NoSuchProviderException.h File Reference	4101
7.661src/main/decaf/security/Principal.h File Reference	4102
7.662src/main/decaf/security/PublicKey.h File Reference	4103
7.663src/main/decaf/security/SignatureException.h File Reference	4104
7.664src/main/decaf/security_provider/SecurityProvider.h File Reference	4105
7.665src/main/decaf/security_provider/SecurityProviderMap.h File Reference	4106
7.666src/main/decaf/security_provider/SecurityProviderRegistrar.h File Reference . . .	4107
7.667src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h File Reference	4108

7.668src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h File Reference	4109
7.669src/main/decaf/util/AbstractCollection.h File Reference	4110
7.670src/main/decaf/util/AbstractList.h File Reference	4111
7.671src/main/decaf/util/AbstractMap.h File Reference	4112
7.672src/main/decaf/util/AbstractQueue.h File Reference	4113
7.673src/main/decaf/util/AbstractSequentialList.h File Reference	4114
7.674src/main/decaf/util/AbstractSet.h File Reference	4115
7.675src/main/decaf/util/Collection.h File Reference	4116
7.676src/main/decaf/util/Comparator.h File Reference	4117
7.677src/main/decaf/util/comparators/Less.h File Reference	4118
7.678src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference	4119
7.679src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference	4120
7.680src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference	4121
7.681src/main/decaf/util/concurrent/BlockingQueue.h File Reference	4122
7.682src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference	4123
7.683src/main/decaf/util/concurrent/Callable.h File Reference	4124
7.684src/main/decaf/util/concurrent/CancellationException.h File Reference	4125
7.685src/main/decaf/util/concurrent/Concurrent.h File Reference	4126
7.685.1 Define Documentation	4126
7.685.1.1 synchronized	4126
7.685.1.2 WAIT_INFINITE	4126
7.686src/main/decaf/util/concurrent/ConcurrentMap.h File Reference	4127
7.687src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference	4128
7.688src/main/decaf/util/concurrent/CountDownLatch.h File Reference	4129
7.689src/main/decaf/util/concurrent/Delayed.h File Reference	4130
7.690src/main/decaf/util/concurrent/ExecutionException.h File Reference	4131
7.691src/main/decaf/util/concurrent/Executor.h File Reference	4132
7.692src/main/decaf/util/concurrent/ExecutorService.h File Reference	4133
7.693src/main/decaf/util/concurrent/Future.h File Reference	4134
7.694src/main/decaf/util/concurrent/Lock.h File Reference	4135
7.695src/main/decaf/util/concurrent/locks/Lock.h File Reference	4136
7.696src/main/decaf/util/concurrent/locks/Condition.h File Reference	4137
7.697src/main/decaf/util/concurrent/locks/LockSupport.h File Reference	4138
7.698src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference	4139
7.699src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference	4140
7.700src/main/decaf/util/concurrent/Mutex.h File Reference	4141

7.701src/main/decaf/util/concurrent/PooledThread.h File Reference	4142
7.702src/main/decaf/util/concurrent/PooledThreadListener.h File Reference	4143
7.703src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference	4144
7.704src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference	4145
7.705src/main/decaf/util/concurrent/Semaphore.h File Reference	4146
7.706src/main/decaf/util/concurrent/Synchronizable.h File Reference	4147
7.707src/main/decaf/util/concurrent/SynchronousQueue.h File Reference	4148
7.708src/main/decaf/util/concurrent/TaskListener.h File Reference	4149
7.709src/main/decaf/util/concurrent/ThreadFactory.h File Reference	4150
7.710src/main/decaf/util/concurrent/ThreadPool.h File Reference	4151
7.711src/main/decaf/util/concurrent/TimeoutException.h File Reference	4152
7.712src/main/decaf/util/concurrent/TimeUnit.h File Reference	4153
7.713src/main/decaf/util/Date.h File Reference	4154
7.714src/main/decaf/util/Iterator.h File Reference	4155
7.715src/main/decaf/util/List.h File Reference	4156
7.716src/main/decaf/util/ListIterator.h File Reference	4157
7.717src/main/decaf/util/logging/ConsoleHandler.h File Reference	4158
7.718src/main/decaf/util/logging/Filter.h File Reference	4159
7.719src/main/decaf/util/logging/Formatter.h File Reference	4160
7.720src/main/decaf/util/logging/Handler.h File Reference	4161
7.721src/main/decaf/util/logging/Logger.h File Reference	4162
7.722src/main/decaf/util/logging/LoggerCommon.h File Reference	4163
7.723src/main/decaf/util/logging/LoggerDefines.h File Reference	4164
7.723.1 Define Documentation	4164
7.723.1.1 LOGDECAF_DEBUG	4164
7.723.1.2 LOGDECAF_DEBUG_1	4164
7.723.1.3 LOGDECAF_DECLARE	4165
7.723.1.4 LOGDECAF_DECLARE_LOCAL	4165
7.723.1.5 LOGDECAF_ERROR	4165
7.723.1.6 LOGDECAF_FATAL	4165
7.723.1.7 LOGDECAF_INFO	4165
7.723.1.8 LOGDECAF_INITIALIZE	4165
7.723.1.9 LOGDECAF_WARN	4165
7.724src/main/decaf/util/logging/LoggerHierarchy.h File Reference	4166
7.725src/main/decaf/util/logging/LogManager.h File Reference	4167
7.726src/main/decaf/util/logging/LogRecord.h File Reference	4168

7.727	src/main/decaf/util/logging/LogWriter.h File Reference	4169
7.728	src/main/decaf/util/logging/MarkBlockLogger.h File Reference	4170
7.729	src/main/decaf/util/logging/PropertiesChangeListener.h File Reference	4171
7.730	src/main/decaf/util/logging/SimpleFormatter.h File Reference	4172
7.731	src/main/decaf/util/logging/SimpleLogger.h File Reference	4173
7.732	src/main/decaf/util/logging/StreamHandler.h File Reference	4174
7.733	src/main/decaf/util/Map.h File Reference	4175
7.734	src/main/decaf/util/PriorityQueue.h File Reference	4176
7.735	src/main/decaf/util/Properties.h File Reference	4177
7.736	src/main/decaf/util/Random.h File Reference	4178
7.737	src/main/decaf/util/Set.h File Reference	4179
7.738	src/main/decaf/util/StlList.h File Reference	4180
7.739	src/main/decaf/util/StlMap.h File Reference	4181
7.740	src/main/decaf/util/StlQueue.h File Reference	4182
7.741	src/main/decaf/util/StlSet.h File Reference	4183
7.742	src/main/decaf/util/StringTokenizer.h File Reference	4184
7.743	src/main/decaf/util/Timer.h File Reference	4185
7.744	src/main/decaf/util/TimerTask.h File Reference	4186
7.745	src/main/decaf/util/UUID.h File Reference	4187

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

activemq (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	69
activemq::cmsutil	70
activemq::commands	71
activemq::core	73
activemq::exceptions	74
activemq::io	75
activemq::library	76
activemq::state	77
activemq::threads	78
activemq::transport	79
activemq::transport::correlator	80
activemq::transport::failover	81
activemq::transport::inactivity	82
activemq::transport::logging	83
activemq::transport::mock	84
activemq::transport::tcp	85
activemq::util	86
activemq::wireformat	87
activemq::wireformat::openwire	88
activemq::wireformat::openwire::marshal	89
activemq::wireformat::openwire::marshal::v1	90
activemq::wireformat::openwire::marshal::v2	94
activemq::wireformat::openwire::marshal::v3	98
activemq::wireformat::openwire::marshal::v4	102
activemq::wireformat::openwire::marshal::v5	106
activemq::wireformat::openwire::utils	110
activemq::wireformat::stomp	111
cms (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	112
decaf (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	115
decaf::internal	116

<code>decaf::internal::io</code>	117
<code>decaf::internal::net</code>	118
<code>decaf::internal::nio</code>	119
<code>decaf::internal::util</code>	120
<code>decaf::internal::util::concurrent</code>	121
<code>decaf::io</code>	122
<code>decaf::lang</code>	124
<code>decaf::lang::exceptions</code>	126
<code>decaf::net</code>	127
<code>decaf::nio</code>	128
<code>decaf::security</code>	129
<code>decaf::security::auth</code>	130
<code>decaf::security::auth::x500</code>	131
<code>decaf::security::cert</code>	132
<code>decaf::security_provider</code>	133
<code>decaf::security_provider::unix</code>	134
<code>decaf::security_provider::unix::openssl</code>	135
<code>decaf::util</code>	136
<code>decaf::util::comparators</code>	138
<code>decaf::util::concurrent</code>	139
<code>decaf::util::concurrent::atomic</code>	141
<code>decaf::util::concurrent::locks</code>	142
<code>decaf::util::logging</code>	143
<code>std</code>	145

Chapter 2

Data Structure Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

activemq::core::ActiveMQAckHandler	171
activemq::core::ActiveMQConstants	260
activemq::library::ActiveMQCPP	272
activemq::core::ActiveMQTransactionContext	622
decaf::lang::Appendable	626
decaf::nio::CharBuffer	998
decaf::internal::nio::CharArrayBuffer	990
decaf::internal::AprPool	628
decaf::util::concurrent::atomic::AtomicBoolean	630
decaf::lang::AtomicRefCounter	638
decaf::util::concurrent::atomic::AtomicReference< T >	641
binary_function	720
decaf::util::Comparator< E >	1086
decaf::util::comparators::Less< E >	1981
decaf::util::Comparator< Pointer< T, R > >	1086
decaf::lang::PointerComparator< T, R >	2504
std::less< decaf::lang::Pointer< T > >	1983
activemq::wireformat::openwire::utils::BooleanStream	739
decaf::nio::Buffer	803
decaf::nio::ByteBuffer	913
decaf::internal::nio::ByteArrayBuffer	870
decaf::nio::CharBuffer	998
decaf::nio::DoubleBuffer	1556
decaf::internal::nio::DoubleArrayBuffer	1548
decaf::nio::FloatBuffer	1665
decaf::internal::nio::FloatArrayBuffer	1658
decaf::nio::IntBuffer	1751
decaf::internal::nio::IntArrayBuffer	1744
decaf::nio::LongBuffer	2079
decaf::internal::nio::LongArrayBuffer	2071
decaf::nio::ShortBuffer	2923
decaf::internal::nio::ShortArrayBuffer	2915

decaf::internal::nio::BufferFactory	824
decaf::internal::util::ByteArrayAdapter	849
decaf::internal::nio::ByteArrayPerspective	908
decaf::util::concurrent::Callable< V >	964
decaf::security::cert::Certificate	968
decaf::security::cert::X509Certificate	3407
decaf::security_provider::unix::openssl::OpenSSLX509Certificate	2442
decaf::lang::CharSequence	1014
decaf::nio::CharBuffer	998
decaf::io::Closeable	1019
activemq::transport::Transport	3273
activemq::transport::CompositeTransport	1095
activemq::transport::failover::FailoverTransport	1612
activemq::transport::IOTransport	1823
activemq::transport::mock::MockTransport	2371
activemq::transport::TransportFilter	3281
activemq::transport::correlator::ResponseCorrelator	2787
activemq::transport::inactivity::InactivityMonitor	1733
activemq::transport::logging::LoggingTransport	2042
activemq::transport::tcp::TcpTransport	3160
activemq::wireformat::WireFormatNegotiator	3399
activemq::wireformat::openwire::OpenWireFormatNegotiator	2462
decaf::io::InputStream	1740
decaf::internal::io::StandardInputStream	3000
decaf::io::BlockingByteArrayInputStream	724
decaf::io::ByteArrayInputStream	892
decaf::io::FilterInputStream	1631
activemq::io::LoggingInputStream	2038
decaf::io::BufferedInputStream	809
decaf::io::DataInputStream	1379
decaf::net::SocketInputStream	2977
decaf::io::OutputStream	2470
decaf::internal::io::StandardErrorOutputStream	2994
decaf::internal::io::StandardOutputStream	3008
decaf::io::ByteArrayOutputStream	900
decaf::io::FilterOutputStream	1640
activemq::io::LoggingOutputStream	2040
decaf::io::BufferedOutputStream	814
decaf::io::DataOutputStream	1388
decaf::net::SocketOutputStream	2984
decaf::net::Socket	2964
decaf::net::BufferedSocket	817
decaf::net::TcpSocket	3152
decaf::util::logging::Handler	1709
decaf::util::logging::StreamHandler	3077
cms::Closeable	1021
activemq::commands::ActiveMQTempDestination	499
activemq::commands::ActiveMQTempQueue	523
activemq::commands::ActiveMQTempTopic	548
cms::Connection	1131
activemq::core::ActiveMQConnection	233

cms::MessageConsumer	2214
activemq::cmsutil::CachedConsumer	953
activemq::core::ActiveMQConsumer	263
cms::MessageProducer	2332
activemq::cmsutil::CachedProducer	957
activemq::core::ActiveMQProducer	406
cms::QueueBrowser	2675
cms::Session	2839
activemq::cmsutil::PooledSession	2505
activemq::core::ActiveMQSession	443
activemq::cmsutil::CmsAccessor	1024
activemq::cmsutil::CmsDestinationAccessor	1028
activemq::cmsutil::CmsTemplate	1041
cms::CMSException	1031
cms::CMSSecurityException	1040
cms::IllegalStateException	1729
cms::InvalidClientIdException	1808
cms::InvalidDestinationException	1809
cms::InvalidSelectorException	1816
cms::MessageEOFException	2277
cms::MessageFormatException	2278
cms::MessageNotReadableException	2330
cms::MessageNotWriteableException	2331
cms::UnsupportedOperationException	3306
activemq::util::CMSExceptionSupport	1034
cms::CMSProperties	1036
activemq::util::ActiveMQProperties	414
activemq::state::CommandVisitor	1069
activemq::state::CommandVisitorAdapter	1077
activemq::state::ConnectionStateTracker	1244
decaf::lang::Comparable< T >	1083
decaf::lang::Comparable< bool >	1083
decaf::lang::Boolean	732
decaf::lang::Comparable< Boolean >	1083
decaf::lang::Boolean	732
decaf::lang::Comparable< BrokerId >	1083
activemq::commands::BrokerId	751
decaf::lang::Comparable< Byte >	1083
decaf::lang::Byte	840
decaf::lang::Comparable< ByteBuffer >	1083
decaf::nio::ByteBuffer	913
decaf::lang::Comparable< char >	1083
decaf::lang::Character	982
decaf::lang::Comparable< Character >	1083
decaf::lang::Character	982
decaf::lang::Comparable< CharBuffer >	1083
decaf::nio::CharBuffer	998
decaf::lang::Comparable< ConnectionId >	1083
activemq::commands::ConnectionId	1187

decaf::lang::Comparable< ConsumerId >	1083
activemq::commands::ConsumerId	1275
decaf::lang::Comparable< Date >	1083
decaf::util::Date	1467
decaf::lang::Comparable< Delayed >	1083
decaf::util::concurrent::Delayed	1477
decaf::lang::Comparable< Double >	1083
decaf::lang::Double	1537
decaf::lang::Comparable< double >	1083
decaf::lang::Double	1537
decaf::lang::Comparable< DoubleBuffer >	1083
decaf::nio::DoubleBuffer	1556
decaf::lang::Comparable< float >	1083
decaf::lang::Float	1647
decaf::lang::Comparable< Float >	1083
decaf::lang::Float	1647
decaf::lang::Comparable< FloatBuffer >	1083
decaf::nio::FloatBuffer	1665
decaf::lang::Comparable< int >	1083
decaf::lang::Integer	1762
decaf::lang::Comparable< IntBuffer >	1083
decaf::nio::IntBuffer	1751
decaf::lang::Comparable< Integer >	1083
decaf::lang::Integer	1762
decaf::lang::Comparable< LocalTransactionId >	1083
activemq::commands::LocalTransactionId	1993
decaf::lang::Comparable< Long >	1083
decaf::lang::Long	2057
decaf::lang::Comparable< long long >	1083
decaf::lang::Long	2057
decaf::lang::Comparable< LongBuffer >	1083
decaf::nio::LongBuffer	2079
decaf::lang::Comparable< MessageId >	1083
activemq::commands::MessageId	2279
decaf::lang::Comparable< ProducerId >	1083
activemq::commands::ProducerId	2606
decaf::lang::Comparable< SessionId >	1083
activemq::commands::SessionId	2853
decaf::lang::Comparable< Short >	1083
decaf::lang::Short	2906
decaf::lang::Comparable< short >	1083
decaf::lang::Short	2906
decaf::lang::Comparable< ShortBuffer >	1083
decaf::nio::ShortBuffer	2923
decaf::lang::Comparable< TimeUnit >	1083
decaf::util::concurrent::TimeUnit	3205

decaf::lang::Comparable< TransactionId >	1083
activemq::commands::TransactionId	3217
activemq::commands::LocalTransactionId	1993
activemq::commands::XATransactionId	3410
decaf::lang::Comparable< unsigned char >	1083
decaf::lang::Byte	840
decaf::lang::Comparable< URI >	1083
decaf::net::URI	3308
decaf::lang::Comparable< UUID >	1083
decaf::util::UUID	3356
decaf::lang::Comparable< XATransactionId >	1083
activemq::commands::XATransactionId	3410
decaf::util::Comparator< T >	1086
activemq::util::CompositeData	1088
decaf::util::concurrent::locks::Condition	1118
decaf::util::concurrent::ConditionHandle	1124
decaf::internal::util::concurrent::ConditionImpl	1126
cms::ConnectionFactory	1184
activemq::core::ActiveMQConnectionFactory	244
cms::ConnectionMetaData	1237
activemq::core::ActiveMQConnectionMetaData	249
activemq::state::ConnectionState	1241
activemq::state::ConsumerState	1329
decaf::util::concurrent::CountDownLatch	1354
activemq::wireformat::openwire::marshal::DataStreamMarshaller	1418
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	692
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	289
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller	427
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	507
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	532
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	557
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	606
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	664
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	787
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	1144
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	1168
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1225
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	1263
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1313
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	1342
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1501
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	1683
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	1949
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2210
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2243
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2269
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	2325
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	181
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	217
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	326
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller	349

activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	390
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	483
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	582
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2359
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	2599
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2648
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	2707
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	2740
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	2772
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	2806
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller	1367
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	1402
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1589
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	1784
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	2881
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	2937
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3245
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	759
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	1195
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	1288
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	1531
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	1850
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	1867
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller	1898
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	1922
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2300
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	2409
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2489
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	1965
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2623
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	2861
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3102
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3220
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	2009
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	3419
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3395
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	301
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	439
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	519
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	544
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	569
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	618
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	685
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	799
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	1156
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	1180
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	1233
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	1271
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1325
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	1350
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1489
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	1679
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	1933

activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2194
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2231
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	2257
activemq::wireformat::openwire::marshal::v2::MessageMarshaller	2305
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	193
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	229
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	338
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	361
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	402
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	495
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	594
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2351
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2583
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2636
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2711
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	2736
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	2756
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	2791
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	1375
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	1414
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1585
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	1780
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	2889
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	2949
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3261
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	771
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1207
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	1296
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	1515
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	1838
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	1863
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller	1886
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	1910
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2284
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	2397
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2477
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	1961
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2611
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	2865
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3118
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3224
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	1997
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	3427
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3387
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	285
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	423
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	503
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	528
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	553
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	602
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	657
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	783
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1140

activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1164
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1217
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	1255
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1309
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	1334
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1493
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	1687
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	1941
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2202
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2239
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	2265
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	2315
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	177
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	213
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	322
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller .	345
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	386
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	479
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	578
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2355
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	2587
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	2652
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	2715
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	2744
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	2760
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	2796
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller	1359
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1398
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1593
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller . .	1788
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	2897
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	2945
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3253
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	755
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1191
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1280
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	1519
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	1846
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	1871
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	1894
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	1914
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2292
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	2401
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2481
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller .	1969
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	2615
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	2869
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3110
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3236
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller . .	2001
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller . . .	3415
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	3379
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	293

activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	431
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	511
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	536
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	561
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	610
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	671
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	791
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller . .	1148
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller . . .	1172
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1221
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller . . .	1259
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1317
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller . . .	1338
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1497
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1691
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	1945
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2206
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller . . .	2235
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	2261
activemq::wireformat::openwire::marshal::v4::MessageMarshaller	2310
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	185
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	221
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	330
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller .	353
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	394
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	487
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	586
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2367
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2591
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2644
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	2719
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	2748
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller . . .	2768
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	2811
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	1363
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller . . .	1410
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1597
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller .	1792
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	2885
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	2941
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3249
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	763
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1199
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1284
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	1523
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	1842
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	1875
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	1890
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	1918
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2288
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller . . .	2413
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2485
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	1973

activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2627
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	2873
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	3106
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	3228
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller . . .	2005
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller . . .	3423
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	3391
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller . . .	297
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	435
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	515
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	540
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	565
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	614
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	678
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	795
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller . .	1152
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller . . .	1176
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	1229
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller . . .	1267
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	1321
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller . . .	1346
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	1505
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1695
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	1937
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2198
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller . . .	2247
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	2273
activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2320
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	189
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	225
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	334
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller .	357
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	398
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	491
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	590
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2363
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	2595
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	2640
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	2723
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	2732
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller . . .	2764
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	2801
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1371
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller . . .	1406
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1601
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller . .	1796
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	2893
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	2953
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	3257
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	767
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	1203
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	1292
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller	1527

activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	1854
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	1879
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	1902
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	1926
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2296
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	2405
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2493
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	1977
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2619
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	2857
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	3114
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	3232
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	2013
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	3431
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	3383
cms::DeliveryMode	1478
cms::Destination	1480
cms::Queue	2674
activemq::commands::ActiveMQQueue	419
cms::TemporaryQueue	3166
activemq::commands::ActiveMQTempQueue	523
cms::TemporaryTopic	3168
activemq::commands::ActiveMQTempTopic	548
cms::Topic	3215
activemq::commands::ActiveMQTopic	598
activemq::commands::ActiveMQDestination::DestinationFilter	1483
activemq::cmsutil::DestinationResolver	1509
activemq::cmsutil::DynamicDestinationResolver	1568
activemq::core::DispatchData	1535
activemq::core::Dispatcher	1536
activemq::core::ActiveMQConsumer	263
activemq::core::ActiveMQSession	443
decaf::lang::DYNAMIC_CAST_TOKEN	1567
decaf::util::Map< K, V, COMPARATOR >::Entry	1570
cms::ExceptionListener	1581
decaf::util::concurrent::Executor	1608
decaf::util::concurrent::ExecutorService	1610
decaf::util::logging::Filter	1630
decaf::util::logging::Formatter	1699
decaf::util::logging::SimpleFormatter	2960
decaf::util::concurrent::Future< V >	1701
activemq::transport::correlator::FutureResponse	1704
decaf::security::GeneralSecurityException	1706
decaf::security::cert::CertificateException	974
decaf::security::cert::CertificateEncodingException	972
decaf::security::cert::CertificateExpiredException	976
decaf::security::cert::CertificateNotYetValidException	978
decaf::security::cert::CertificateParsingException	980
decaf::security::KeyException	1955
decaf::security::InvalidKeyException	1810
decaf::security::NoSuchAlgorithmExceptionException	2420

decaf::security::NoSuchProviderException	2426
decaf::security::SignatureException	2957
decaf::internal::util::HexStringParser	1713
activemq::wireformat::openwire::utils::HexTable	1715
decaf::lang::Iterable< E >	1830
decaf::util::Collection< E >	1054
decaf::util::AbstractCollection< E >	147
decaf::util::List< E >	1984
decaf::util::AbstractList< E >	160
decaf::util::AbstractSequentialList< E >	166
decaf::util::StlList< E >	3017
decaf::util::Queue< E >	2671
decaf::util::AbstractQueue< E >	162
decaf::util::PriorityQueue< E >	2571
decaf::util::Set< E >	2905
decaf::util::AbstractSet< E >	167
decaf::util::StlSet< E >	3051
decaf::lang::Iterable< PrimitiveValueNode >	1830
decaf::util::Collection< PrimitiveValueNode >	1054
decaf::util::AbstractCollection< PrimitiveValueNode >	147
decaf::util::List< PrimitiveValueNode >	1984
decaf::util::StlList< PrimitiveValueNode >	3017
activemq::util::PrimitiveList	2525
decaf::util::Iterator< T >	1832
decaf::util::Iterator< E >	1832
decaf::util::ListIterator< E >	1990
decaf::security::Key	1953
decaf::security::PublicKey	2670
decaf::util::concurrent::locks::Lock	2017
decaf::util::concurrent::locks::ReentrantLock	2692
decaf::util::concurrent::Lock	2023
decaf::util::concurrent::locks::LockSupport	2025
decaf::util::logging::Logger	2028
decaf::util::logging::LoggerHierarchy	2037
decaf::util::logging::LogManager	2045
decaf::util::logging::LogRecord	2050
decaf::util::logging::LogWriter	2055
activemq::util::LongSequenceGenerator	2090
decaf::util::logging::MarkBlockLogger	2117
activemq::wireformat::MarshalAware	2118
activemq::commands::DataStructure	1461
activemq::commands::BaseDataStructure	715
activemq::commands::ActiveMQDestination	274
activemq::commands::ActiveMQQueue	419
activemq::commands::ActiveMQTempDestination	499
activemq::commands::ActiveMQTopic	598
activemq::commands::BooleanExpression	737
activemq::commands::BrokerId	751
activemq::commands::Command	1063
activemq::commands::BaseCommand	650

activemq::commands::BrokerError	745
activemq::commands::BrokerInfo	775
activemq::commands::ConnectionControl	1135
activemq::commands::ConnectionError	1160
activemq::commands::ConnectionInfo	1211
activemq::commands::ConsumerControl	1250
activemq::commands::ConsumerInfo	1300
activemq::commands::ControlCommand	1330
activemq::commands::DestinationInfo	1484
activemq::commands::FlushCommand	1676
activemq::commands::KeepAliveInfo	1930
activemq::commands::Message	2145
activemq::commands::ActiveMQMessageTemplate< T >	365
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >	365
activemq::commands::ActiveMQBytesMessage	197
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >	365
activemq::commands::ActiveMQMapMessage	308
activemq::commands::ActiveMQMessageTemplate< cms::Message >	365
activemq::commands::ActiveMQBlobMessage	172
activemq::commands::ActiveMQMessage	342
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >	365
activemq::commands::ActiveMQObjectMessage	383
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >	365
activemq::commands::ActiveMQStreamMessage	464
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >	365
activemq::commands::ActiveMQTextMessage	573
activemq::commands::MessageAck	2188
activemq::commands::MessageDispatch	2219
activemq::commands::MessageDispatchNotification	2251
activemq::commands::MessagePull	2346
activemq::commands::ProducerAck	2579
activemq::commands::ProducerInfo	2631
activemq::commands::RemoveInfo	2703
activemq::commands::RemoveSubscriptionInfo	2727
activemq::commands::ReplayCommand	2752
activemq::commands::Response	2781
activemq::commands::DataArrayResponse	1356
activemq::commands::DataResponse	1395
activemq::commands::ExceptionResponse	1582
activemq::commands::IntegerResponse	1777
activemq::state::Tracked	3216
activemq::commands::SessionInfo	2877
activemq::commands::ShutdownInfo	2934
activemq::commands::TransactionInfo	3240
activemq::commands::WireFormatInfo	3369
activemq::commands::ConnectionId	1187
activemq::commands::ConsumerId	1275
activemq::commands::DiscoveryEvent	1511
activemq::commands::JournalQueueAck	1834
activemq::commands::JournalTopicAck	1858
activemq::commands::JournalTrace	1883
activemq::commands::JournalTransaction	1906

activemq::commands::MessageId	2279
activemq::commands::NetworkBridgeFilter	2393
activemq::commands::PartialCommand	2473
activemq::commands::LastPartialCommand	1958
activemq::commands::ProducerId	2606
activemq::commands::SessionId	2853
activemq::commands::SubscriptionInfo	3097
activemq::commands::TransactionId	3217
activemq::wireformat::openwire::marshal::v2::MarshallerFactory	2121
activemq::wireformat::openwire::marshal::v4::MarshallerFactory	2122
activemq::wireformat::openwire::marshal::v1::MarshallerFactory	2123
activemq::wireformat::openwire::marshal::v3::MarshallerFactory	2124
activemq::wireformat::openwire::marshal::v5::MarshallerFactory	2125
decaf::lang::Math	2126
cms::Message	2163
activemq::commands::ActiveMQMessageTemplate< cms::Message >	365
cms::BytesMessage	938
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >	365
cms::MapMessage	2106
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >	365
cms::ObjectMessage	2438
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >	365
cms::StreamMessage	3081
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >	365
cms::TextMessage	3170
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >	365
activemq::cmsutil::MessageCreator	2218
cms::MessageListener	2304
activemq::wireformat::openwire::utils::MessagePropertyInterceptor	2339
decaf::util::concurrent::MutexHandle	2390
decaf::internal::util::concurrent::MutexImpl	2391
decaf::lang::Number	2432
decaf::lang::Byte	840
decaf::lang::Character	982
decaf::lang::Double	1537
decaf::lang::Float	1647
decaf::lang::Integer	1762
decaf::lang::Long	2057
decaf::lang::Short	2906
decaf::util::concurrent::atomic::AtomicInteger	633
decaf::security_provider::unix::openssl::OpenSSLX500Principal	2439
activemq::wireformat::openwire::utils::OpenwireStringSupport	2468
decaf::lang::Pointer< T, REFCOUNTER >	2497
decaf::util::concurrent::PooledThread	2518
decaf::util::concurrent::PooledThreadListener	2520
decaf::util::concurrent::ThreadPool	3175
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller	2546
activemq::util::PrimitiveValueNode::PrimitiveValue	2551
activemq::util::PrimitiveValueConverter	2553
activemq::util::PrimitiveValueNode	2555
decaf::security::Principal	2569
decaf::security::auth::x500::X500Principal	3406

activemq::cmsutil::ProducerCallback	2603
activemq::cmsutil::CmsTemplate::SendExecutor	2836
activemq::state::ProducerState	2656
decaf::util::Properties	2657
decaf::util::logging::PropertiesChangeListener	2666
decaf::util::Random	2677
decaf::io::Reader	2683
decaf::util::concurrent::locks::ReadWriteLock	2688
decaf::util::concurrent::RejectedExecutionHandler	2702
activemq::cmsutil::ResourceLifecycleManager	2778
activemq::transport::mock::ResponseBuilder	2785
activemq::wireformat::openwire::OpenWireResponseBuilder	2466
decaf::lang::Runnable	2816
activemq::threads::CompositeTaskRunner	1092
activemq::threads::DedicatedTaskRunner	1473
activemq::transport::IOTransport	1823
decaf::util::TimerTask	3199
activemq::transport::inactivity::ReadChecker	2682
activemq::transport::inactivity::WriteChecker	3403
decaf::lang::Runtime	2817
decaf::internal::DecafRuntime	1472
decaf::security_provider::SecurityProvider	2822
decaf::security_provider::SecurityProviderMap	2823
decaf::security_provider::SecurityProviderRegistrar	2825
decaf::util::concurrent::Semaphore	2827
decaf::net::ServerSocket	2837
activemq::cmsutil::SessionCallback	2852
activemq::cmsutil::CmsTemplate::ProducerExecutor	2604
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	2776
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2690
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	2777
activemq::cmsutil::SessionPool	2901
activemq::state::SessionState	2903
decaf::util::logging::SimpleLogger	2962
decaf::net::SocketError	2971
decaf::net::SocketFactory	2975
activemq::commands::BrokerError::StackTraceElement	2993
cms::Startable	3014
cms::Connection	1131
decaf::lang::STATIC_CAST_TOKEN	3015
activemq::core::ActiveMQConstants::StaticInitializer	3016
activemq::wireformat::stomp::StompCommandConstants	3057
activemq::wireformat::stomp::StompFrame	3061
activemq::wireformat::stomp::StompHelper	3066
cms::Stoppable	3076
cms::Connection	1131
decaf::util::StringTokenizer	3094
decaf::util::concurrent::Synchronizable	3122
activemq::core::MessageDispatchChannel	2224
decaf::util::Collection< PrimitiveValueNode >	1054
decaf::internal::util::concurrent::SynchronizableImpl	3133

decaf::io::InputStream	1740
decaf::io::OutputStream	2470
decaf::util::Collection< E >	1054
decaf::util::concurrent::Mutex	2385
decaf::util::Map< K, V, COMPARATOR >	2094
decaf::util::AbstractMap< K, V, COMPARATOR >	161
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >	1097
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >	1102
decaf::util::StlMap< K, V, COMPARATOR >	3030
decaf::util::StlQueue< T >	3043
decaf::util::Map< std::string, PrimitiveValueNode, std::less< std::string > >	2094
decaf::util::StlMap< std::string, PrimitiveValueNode >	3030
activemq::util::PrimitiveMap	2536
activemq::core::Synchronization	3137
decaf::util::concurrent::SynchronousQueue< E >	3138
decaf::lang::System	3145
activemq::threads::Task	3148
activemq::core::ActiveMQSessionExecutor	460
activemq::threads::CompositeTask	1090
activemq::transport::failover::BackupTransportPool	647
activemq::transport::failover::CloseTransportsTask	1022
activemq::transport::failover::FailoverTransport	1612
activemq::threads::CompositeTaskRunner	1092
decaf::util::concurrent::TaskListener	3149
activemq::threads::TaskRunner	3150
activemq::threads::CompositeTaskRunner	1092
activemq::threads::DedicatedTaskRunner	1473
decaf::util::concurrent::ThreadFactory	3172
decaf::lang::ThreadGroup	3174
decaf::lang::Throwable	3181
decaf::lang::Exception	1574
activemq::exceptions::ActiveMQException	305
activemq::exceptions::BrokerException	749
decaf::io::IOException	1820
decaf::io::EOFException	1571
decaf::io::InterruptedIOException	1805
decaf::net::SocketTimeoutException	2990
decaf::io::UnsupportedEncodingException	3300
decaf::io::UTFDataFormatException	3353
decaf::net::HttpRetryException	1717
decaf::net::MalformedURLException	2091
decaf::net::ProtocolException	2667
decaf::net::SocketException	2972
decaf::net::BindException	721
decaf::net::ConnectException	1128
decaf::net::NoRouteToHostException	2417
decaf::net::PortUnreachableException	2522
decaf::net::UnknownHostException	3294
decaf::net::UnknownServiceException	3297
decaf::lang::exceptions::ClassCastException	1016
decaf::lang::exceptions::IllegalArgumentException	1720
decaf::lang::exceptions::IllegalMonitorStateException	1723

decaf::lang::exceptions::IllegalStateException	1726
decaf::nio::InvalidMarkException	1813
decaf::lang::exceptions::IllegalThreadStateException	1730
decaf::lang::exceptions::IndexOutOfBoundsException	1737
decaf::lang::exceptions::InterruptedException	1802
decaf::lang::exceptions::InvalidStateException	1817
decaf::lang::exceptions::NoSuchElementException	2423
decaf::lang::exceptions::NullPointerException	2429
decaf::lang::exceptions::NumberFormatException	2435
decaf::lang::exceptions::RuntimeException	2819
decaf::lang::exceptions::UnsupportedOperationException	3303
decaf::nio::ReadOnlyBufferException	2685
decaf::net::URISyntaxException	3335
decaf::nio::BufferOverflowException	834
decaf::nio::BufferUnderflowException	837
decaf::util::concurrent::BrokenBarrierException	742
decaf::util::concurrent::CancellationException	965
decaf::util::concurrent::ExecutionException	1605
decaf::util::concurrent::RejectedExecutionException	2699
decaf::util::concurrent::TimeoutException	3185
decaf::util::Timer	3188
decaf::internal::util::TimerTaskHeap	3202
activemq::state::TransactionState	3265
decaf::internal::util::concurrent::Transferer< E >	3267
decaf::internal::util::concurrent::TransferQueue< E >	3268
decaf::internal::util::concurrent::TransferStack< E >	3271
activemq::transport::TransportFactory	3279
activemq::transport::AbstractTransportFactory	169
activemq::transport::failover::FailoverTransportFactory	1624
activemq::transport::mock::MockTransportFactory	2382
activemq::transport::tcp::TcpTransportFactory	3163
activemq::transport::TransportListener	3289
activemq::core::ActiveMQConnectionSupport	253
activemq::core::ActiveMQConnection	233
activemq::transport::DefaultTransportListener	1475
activemq::transport::failover::BackupTransport	644
activemq::transport::mock::InternalCommandListener	1800
activemq::transport::failover::FailoverTransportListener	1627
activemq::transport::TransportFilter	3281
activemq::transport::TransportRegistry	3291
decaf::internal::net::URIEncoderDecoder	3319
decaf::internal::net::URIHelper	3322
activemq::transport::failover::URIPool	3329
activemq::util::URISupport	3332
decaf::internal::net::URIType	3339
decaf::net::URL	3347
decaf::net::URLDecoder	3349
decaf::net::URLEncoder	3350
activemq::util::Usage	3351
activemq::util::MemoryUsage	2141
activemq::wireformat::WireFormat	3363
activemq::wireformat::openwire::OpenWireFormat	2448

activemq::wireformat::stomp::StompWireFormat	3071
activemq::wireformat::WireFormatFactory	3367
activemq::wireformat::openwire::OpenWireFormatFactory	2460
activemq::wireformat::stomp::StompWireFormatFactory	3075
activemq::wireformat::WireFormatRegistry	3400
decaf::io::Writer	3404

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

decaf::util::AbstractCollection < E > (This class provides a skeletal implementation of the Collection (p.1054) interface, to minimize the effort required to implement this interface)	147
decaf::util::AbstractList < E > (This class provides a skeletal implementation of the List (p.1984) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array))	160
decaf::util::AbstractMap < K , V , COMPARATOR > (This class provides a skeletal implementation of the Map (p.2094) interface, to minimize the effort required to implement this interface)	161
decaf::util::AbstractQueue < E > (This class provides skeletal implementations of some Queue (p.2671) operations)	162
decaf::util::AbstractSequentialList < E > (This class provides a skeletal implementation of the List (p.1984) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list))	166
decaf::util::AbstractSet < E > (This class provides a skeletal implementation of the Set (p.2905) interface to minimize the effort required to implement this interface)	167
activemq::transport::AbstractTransportFactory (Abstract implementation of the TransportFactory (p.3279) interface, providing the base functionality that's common to most of the TransportFactory (p.3279) instances)	169
activemq::core::ActiveMQAckHandler (Interface class that is used to give CMS Messages an interface to Ack themselves with)	171
activemq::commands::ActiveMQBlobMessage	172
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p.177))	177
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p.181))	181
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p.185))	185

activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 189))	189
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 193))	193
activemq::commands::ActiveMQBytesMessage	197
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 213))	213
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 217))	217
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 221))	221
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 225))	225
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 229))	229
activemq::core::ActiveMQConnection (Concrete connection used for all connectors to the ActiveMQ broker)	233
activemq::core::ActiveMQConnectionFactory	244
activemq::core::ActiveMQConnectionMetaData (This class houses all the various settings and information that is used by an instance of an ActiveMQConnection (p. 233) class)	249
activemq::core::ActiveMQConnectionSupport	253
activemq::core::ActiveMQConstants (Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values)	260
activemq::core::ActiveMQConsumer	263
activemq::library::ActiveMQCPP	272
activemq::commands::ActiveMQDestination	274
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 285))	285
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 289))	289
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 293))	293
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 297))	297
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 301))	301
activemq::exceptions::ActiveMQException	305
activemq::commands::ActiveMQMapMessage	308

activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 322))	322
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 326))	326
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 330))	330
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 334))	334
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 338))	338
activemq::commands::ActiveMQMessage	342
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 345))	345
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 349))	349
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 353))	353
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 357))	357
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 361))	361
activemq::commands::ActiveMQMessageTemplate< T >	365
activemq::commands::ActiveMQObjectMessage	383
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 386))	386
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 390))	390
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 394))	394
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 398))	398
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 402))	402
activemq::core::ActiveMQProducer	406
activemq::util::ActiveMQProperties (Implementation of the CMSProperties interface that delegates to a decaf::util::Properties (p. 2657) object)	414
activemq::commands::ActiveMQQueue	419

activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 423))	423
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 427))	427
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 431))	431
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 435))	435
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 439))	439
activemq::core::ActiveMQSession	443
activemq::core::ActiveMQSessionExecutor (Delegate dispatcher for a single session)	460
activemq::commands::ActiveMQStreamMessage	464
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 479))	479
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 483))	483
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 487))	487
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 491))	491
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 495))	495
activemq::commands::ActiveMQTempDestination	499
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 503))	503
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 507))	507
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 511))	511
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 515))	515
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 519))	519
activemq::commands::ActiveMQTempQueue	523
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 528))	528

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 532))	532
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 536))	536
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 540))	540
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 544))	544
activemq::commands::ActiveMQTempTopic	548
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 553))	553
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 557))	557
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 561))	561
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 565))	565
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 569))	569
activemq::commands::ActiveMQTextMessage	573
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 578))	578
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 582))	582
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 586))	586
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 590))	590
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 594))	594
activemq::commands::ActiveMQTopic	598
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 602))	602
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 606))	606
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 610))	610

activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 614))	614
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 618))	618
activemq::core::ActiveMQTransactionContext (Transaction Management class, hold messages that are to be redelivered upon a request to roll-back)	622
decaf::lang::Appendable	626
decaf::internal::AprPool (Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made)	628
decaf::util::concurrent::atomic::AtomicBoolean (A boolean value that may be up- dated atomically)	630
decaf::util::concurrent::atomic::AtomicInteger (An int value that may be updated atomically)	633
decaf::lang::AtomicRefCounter	638
decaf::util::concurrent::atomic::AtomicReference< T > (An Pointer reference that may be updated atomically)	641
activemq::transport::failover::BackupTransport	644
activemq::transport::failover::BackupTransportPool	647
activemq::commands::BaseCommand	650
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 657))	657
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 664))	664
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 671))	671
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 678))	678
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 685))	685
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller (Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol)	692
activemq::commands::BaseDataStructure	715
binary_function	720
decaf::net::BindException	721
decaf::io::BlockingByteArrayInputStream (This is a blocking version of a byte buffer stream)	724
decaf::lang::Boolean	732
activemq::commands::BooleanExpression	737
activemq::wireformat::openwire::utils::BooleanStream (Manages the writing and reading of boolean data streams to and from a data source such as a DataInput- Stream or DataOutputStream)	739
decaf::util::concurrent::BrokenBarrierException	742
activemq::commands::BrokerError (This class represents an Exception sent from the Broker)	745
activemq::exceptions::BrokerException	749

activemq::commands::BrokerId	751
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 755))	755
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 759))	759
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 763))	763
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 767))	767
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 771))	771
activemq::commands::BrokerInfo	775
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 783))	783
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 787))	787
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 791))	791
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 795))	795
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 799))	799
decaf::nio::Buffer (A container for data of a specific primitive type)	803
decaf::io::BufferedInputStream (A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream)	809
decaf::io::BufferedOutputStream (Wrapper around another output stream that buffers output before writing to the target output stream)	814
decaf::net::BufferedSocket (Buffered Socket (p. 2964) class that wraps a Socket (p. 2964) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes)	817
decaf::internal::nio::BufferFactory (Factory class used by static methods in the decaf::nio (p. 128) package to create the various default version of the NIO interfaces)	824
decaf::nio::BufferOverflowException	834
decaf::nio::BufferUnderflowException	837
decaf::lang::Byte	840
decaf::internal::util::ByteArrayAdapter (This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data)	849
decaf::internal::nio::ByteArrayBuffer (This class defines six categories of operations upon byte buffers:)	870
decaf::io::ByteArrayInputStream (Simple implementation of InputStream (p. 1740) that wraps around an STL Vector <code>std::vector<unsigned char></code>)	892
decaf::io::ByteArrayOutputStream	900
decaf::internal::nio::ByteArrayPerspective (This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective)	908
decaf::nio::ByteBuffer (This class defines six categories of operations upon byte buffers:)	913
cms::BytesMessage (A BytesMessage (p. 938) object is used to send a message containing a stream of unsigned bytes)	938

activemq::cmsutil::CachedConsumer (A cached message consumer contained within a pooled session)	953
activemq::cmsutil::CachedProducer (A cached message producer contained within a pooled session)	957
decaf::util::concurrent::Callable< V > (A task that returns a result and may throw an exception)	964
decaf::util::concurrent::CancellationException	965
decaf::security::cert::Certificate (Base interface for all identity certificates)	968
decaf::security::cert::CertificateEncodingException	972
decaf::security::cert::CertificateException	974
decaf::security::cert::CertificateExpiredException	976
decaf::security::cert::CertificateNotYetValidException	978
decaf::security::cert::CertificateParsingException	980
decaf::lang::Character	982
decaf::internal::nio::CharArrayBuffer	990
decaf::nio::CharBuffer (This class defines four categories of operations upon character buffers:)	998
decaf::lang::CharSequence (A CharSequence (p. 1014) is a readable sequence of char values)	1014
decaf::lang::exceptions::ClassCastException	1016
decaf::io::Closeable (Interface for a class that implements the close method)	1019
cms::Closeable (Interface for a class that implements the close method)	1021
activemq::transport::failover::CloseTransportsTask	1022
activemq::cmsutil::CmsAccessor (Base class for activemq::cmsutil::CmsTemplate (p. 1041) and other CMS-accessing gateway helpers, defining common properties such as the CMS cms::ConnectionFactory (p. 1184) to operate on)	1024
activemq::cmsutil::CmsDestinationAccessor (Extends the CmsAccessor (p. 1024) to add support for resolving destination names)	1028
cms::CMSException (CMS API Exception that is the base for all exceptions thrown from CMS classes)	1031
activemq::util::CMSExceptionSupport	1034
cms::CMSProperties (Interface for a Java-like properties object)	1036
cms::CMSSecurityException (This exception must be thrown when a provider rejects a user name/password submitted by a client)	1040
activemq::cmsutil::CmsTemplate (CmsTemplate (p. 1041) simplifies performing synchronous CMS operations)	1041
decaf::util::Collection< E > (The root interface in the collection hierarchy)	1054
activemq::commands::Command	1063
activemq::state::CommandVisitor (Interface for an Object that can visit the various Command Objects that are sent from and to this client)	1069
activemq::state::CommandVisitorAdapter (Default Implementation of a CommandVisitor (p. 1069) that returns NULL for all calls)	1077
decaf::lang::Comparable< T > (This interface imposes a total ordering on the objects of each class that implements it)	1083
decaf::util::Comparator< T > (A comparison function, which imposes a total ordering on some collection of objects)	1086
activemq::util::CompositeData (Represents a Composite URI)	1088
activemq::threads::CompositeTask (Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 1092))	1090
activemq::threads::CompositeTaskRunner (A Task (p. 3148) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added)	1092
activemq::transport::CompositeTransport (A Composite Transport (p. 3273) is a Transport (p. 3273) implementation that is composed of several Transports)	1095

decaf::util::concurrent::ConcurrentMap < K , V , COMPARATOR > (Interface for a Map (p. 2094) type that provides additional atomic putIfAbsent , remove , and replace methods alongside the already available Map (p. 2094) interface) .	1097
decaf::util::concurrent::ConcurrentStlMap < K , V , COMPARATOR > (Map (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map) . . .	1102
decaf::util::concurrent::locks::Condition (Condition (p. 1118) factors out the Mutex (p. 2385) monitor methods (wait , notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary Lock (p. 2017) implementations)	1118
decaf::util::concurrent::ConditionHandle	1124
decaf::internal::util::concurrent::ConditionImpl	1126
decaf::net::ConnectException	1128
cms::Connection (The client's connection to its provider)	1131
activemq::commands::ConnectionControl	1135
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1140))	1140
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1144))	1144
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1148))	1148
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1152))	1152
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1156))	1156
activemq::commands::ConnectionError	1160
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1164))	1164
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1168))	1168
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1172))	1172
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1176))	1176
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1180))	1180
cms::ConnectionFactory (Defines the interface for a factory that creates connection objects, the Connection (p. 1131) objects returned implement the CMS Connection (p. 1131) interface and hide the CMS Provider specific implementation details behind that interface)	1184
activemq::commands::ConnectionId	1187
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1191))	1191

activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1195))	1195
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1199))	1199
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1203))	1203
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1207))	1207
activemq::commands::ConnectionInfo	1211
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1217))	1217
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1221))	1221
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1225))	1225
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1229))	1229
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1233))	1233
cms::ConnectionMetaData (A ConnectionMetaData (p. 1237) object provides information describing the Connection (p. 1131) object)	1237
activemq::state::ConnectionState	1241
activemq::state::ConnectionStateTracker	1244
activemq::commands::ConsumerControl	1250
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1255))	1255
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1259))	1259
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1263))	1263
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1267))	1267
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1271))	1271
activemq::commands::ConsumerId	1275
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1280))	1280

activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1284))	1284
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1288))	1288
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1292))	1292
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1296))	1296
activemq::commands::ConsumerInfo	1300
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1309))	1309
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1313))	1313
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1317))	1317
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1321))	1321
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1325))	1325
activemq::state::ConsumerState	1329
activemq::commands::ControlCommand	1330
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1334))	1334
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1338))	1338
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1342))	1342
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1346))	1346
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1350))	1350
decaf::util::concurrent::CountDownLatch	1354
activemq::commands::DataArrayResponse	1356
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1359))	1359
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1363))	1363

activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1367))	1367
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1371))	1371
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1375))	1375
decaf::io::DataInputStream (A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way)	1379
decaf::io::DataOutputStream (A data output stream lets an application write prim- itive Java data types to an output stream in a portable way)	1388
activemq::commands::DataResponse	1395
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1398))	1398
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1402))	1402
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1406))	1406
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1410))	1410
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1414))	1414
activemq::wireformat::openwire::marshal::DataStreamMarshaller (Base class for all classes that marshal commands for Openwire)	1418
activemq::commands::DataStructure	1461
decaf::util::Date (Wrapper class around a time value in milliseconds)	1467
decaf::internal::DecafRuntime (Handles APR initialization and termination)	1472
activemq::threads::DedicatedTaskRunner	1473
activemq::transport::DefaultTransportListener	1475
decaf::util::concurrent::Delayed (A mix-in style interface for marking objects that should be acted upon after a given delay)	1477
cms::DeliveryMode (This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages)	1478
cms::Destination (A Destination (p. 1480) object encapsulates a provider-specific ad- dress)	1480
activemq::commands::ActiveMQDestination::DestinationFilter	1483
activemq::commands::DestinationInfo	1484
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1489))	1489
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1493))	1493
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1497))	1497

activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1501))	1501
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1505))	1505
activemq::cmsutil::DestinationResolver (Resolves a CMS destination name to a Destination)	1509
activemq::commands::DiscoveryEvent	1511
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1515))	1515
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1519))	1519
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1523))	1523
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1527))	1527
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1531))	1531
activemq::core::DispatchData (Simple POJO that contains the information neces- sary to route a message to a specified consumer)	1535
activemq::core::Dispatcher (Interface for an object responsible for dispatching mes- sages to consumers)	1536
decaf::lang::Double	1537
decaf::internal::nio::DoubleArrayBuffer	1548
decaf::nio::DoubleBuffer (This class defines four categories of operations upon double buffers:)	1556
decaf::lang::DYNAMIC_CAST_TOKEN	1567
activemq::cmsutil::DynamicDestinationResolver (Resolves a CMS destination name to a Destination)	1568
decaf::util::Map< K, V, COMPARATOR >::Entry	1570
decaf::io::EOFException	1571
decaf::lang::Exception	1574
cms::ExceptionListener (If a CMS provider detects a serious problem, it notifies the client application through an ExceptionListener (p. 1581) that is registered with the Connection (p. 1131))	1581
activemq::commands::ExceptionResponse	1582
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1585))	1585
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1589))	1589
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1593))	1593
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1597))	1597

activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1601))	1601
decaf::util::concurrent::ExecutionException	1605
decaf::util::concurrent::Executor (An object that executes submitted decaf.lang Runnable (p. 2816) tasks)	1608
decaf::util::concurrent::ExecutorService (An Executor (p. 1608) that provides methods to manage termination and methods that can produce a Future (p. 1701) for tracking progress of one or more asynchronous tasks)	1610
activemq::transport::failover::FailoverTransport	1612
activemq::transport::failover::FailoverTransportFactory (Creates an instance of a FailoverTransport (p. 1612))	1624
activemq::transport::failover::FailoverTransportListener (Utility class used by the Transport (p. 3273) to perform the work of responding to events from the active Transport (p. 3273))	1627
decaf::util::logging::Filter (A Filter (p. 1630) can be used to provide fine grain control over what is logged, beyond the control provided by log levels)	1630
decaf::io::FilterInputStream (A FilterInputStream (p. 1631) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality)	1631
decaf::io::FilterOutputStream (This class is the superclass of all classes that filter output streams)	1640
decaf::lang::Float	1647
decaf::internal::nio::FloatArrayBuffer	1658
decaf::nio::FloatBuffer (This class defines four categories of operations upon float buffers:)	1665
activemq::commands::FlushCommand	1676
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1679))	1679
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1683))	1683
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1687))	1687
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1691))	1691
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1695))	1695
decaf::util::logging::Formatter (A Formatter (p. 1699) provides support for formatting LogRecords)	1699
decaf::util::concurrent::Future< V > (A Future (p. 1701) represents the result of an asynchronous computation)	1701
activemq::transport::correlator::FutureResponse (A container that holds a response object)	1704
decaf::security::GeneralSecurityException	1706
decaf::util::logging::Handler (A Handler (p. 1709) object takes log messages from a Logger (p. 2028) and exports them)	1709
decaf::internal::util::HexStringParser	1713
activemq::wireformat::openwire::utils::HexTable (Maps hexadecimal strings to the value of an index into the table, i.e)	1715

decaf::net::HttpRetryException	1717
decaf::lang::exceptions::IllegalArgumentException	1720
decaf::lang::exceptions::IllegalMonitorStateException	1723
decaf::lang::exceptions::IllegalStateException	1726
cms::IllegalStateException (This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation)	1729
decaf::lang::exceptions::IllegalThreadStateException	1730
activemq::transport::inactivity::InactivityMonitor	1733
decaf::lang::exceptions::IndexOutOfBoundsException	1737
decaf::io::InputStream (Base interface for an input stream)	1740
decaf::internal::nio::IntArrayBuffer	1744
decaf::nio::IntBuffer (This class defines four categories of operations upon int buffers:)	1751
decaf::lang::Integer	1762
activemq::commands::IntegerResponse	1777
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1780))	1780
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1784))	1784
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1788))	1788
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1792))	1792
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1796))	1796
activemq::transport::mock::InternalCommandListener (Listens for Commands sent from the MockTransport (p. 2371))	1800
decaf::lang::exceptions::InterruptedException	1802
decaf::io::InterruptedException	1805
cms::InvalidClientIdException (This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider)	1808
cms::InvalidDestinationException (This exception must be thrown when a destination either is not understood by a provider or is no longer valid)	1809
decaf::security::InvalidKeyException	1810
decaf::nio::InvalidMarkException	1813
cms::InvalidSelectorException (This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax)	1816
decaf::lang::exceptions::InvalidStateException	1817
decaf::io::IOException	1820
activemq::transport::IOTransport (Implementation of the Transport (p. 3273) interface that performs marshaling of commands to IO streams)	1823
decaf::lang::Iterable< E > (Implementing this interface allows an object to be cast to an Iterable (p. 1830) type for generic collections API calls)	1830
decaf::util::Iterator< T > (Defines an object that can be used to iterate over the elements of a collection)	1832
activemq::commands::JournalQueueAck	1834

activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1838))	1838
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1842))	1842
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1846))	1846
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1850))	1850
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1854))	1854
activemq::commands::JournalTopicAck	1858
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1863))	1863
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1867))	1867
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1871))	1871
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1875))	1875
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1879))	1879
activemq::commands::JournalTrace	1883
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1886))	1886
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1890))	1890
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1894))	1894
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1898))	1898
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1902))	1902
activemq::commands::JournalTransaction	1906
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1910))	1910
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1914))	1914

activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1918))	1918
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1922))	1922
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1926))	1926
activemq::commands::KeepAliveInfo	1930
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1933))	1933
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1937))	1937
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1941))	1941
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1945))	1945
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1949))	1949
decaf::security::Key (The Key (p. 1953) interface is the top-level interface for all keys)	1953
decaf::security::KeyException	1955
activemq::commands::LastPartialCommand	1958
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1961))	1961
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1965))	1965
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1969))	1969
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1973))	1973
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1977))	1977
decaf::util::comparators::Less< E > (Simple Less (p. 1981) Comparator (p. 1086) that compares to elements to determine if the first is less than the second) . .	1981
std::less< decaf::lang::Pointer< T > > (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc)	1983
decaf::util::List< E > (An ordered collection (also known as a sequence))	1984
decaf::util::ListIterator< E > (An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list)	1990
activemq::commands::LocalTransactionId	1993

activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 1997))	1997
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2001))	2001
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2005))	2005
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2009))	2009
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 2013))	2013
decaf::util::concurrent::locks::Lock (Lock (p. 2017) implementations provide more extensive locking operations than can be obtained using synchronized statements)	2017
decaf::util::concurrent::Lock (A wrapper class around a given synchronization mech- anism that provides automatic release upon destruction)	2023
decaf::util::concurrent::locks::LockSupport (Basic thread blocking primitives for creating locks and other synchronization classes)	2025
decaf::util::logging::Logger	2028
decaf::util::logging::LoggerHierarchy	2037
activemq::io::LoggingInputStream	2038
activemq::io::LoggingOutputStream (OutputStream filter that just logs the data being written)	2040
activemq::transport::logging::LoggingTransport (A transport filter that logs com- mands as they are sent/received)	2042
decaf::util::logging::LogManager (There is a single global LogManager (p. 2045) object that is used to maintain a set of shared state about Loggers and log services)	2045
decaf::util::logging::LogRecord	2050
decaf::util::logging::LogWriter	2055
decaf::lang::Long	2057
decaf::internal::nio::LongArrayBuffer	2071
decaf::nio::LongBuffer (This class defines four categories of operations upon long long buffers:)	2079
activemq::util::LongSequenceGenerator (This class is used to generate a sequence of long long values that are incremented each time a new value is requested) .	2090
decaf::net::MalformedURLException	2091
decaf::util::Map< K, V, COMPARATOR > (Map (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide com- mon functions that do not exist in std::map)	2094
cms::MapMessage (A MapMessage (p. 2106) object is used to send a set of name- value pairs)	2106
decaf::util::logging::MarkBlockLogger (Defines a class that can be used to mark the entry and exit from scoped blocks)	2117
activemq::wireformat::MarshalAware	2118
activemq::wireformat::openwire::marshal::v2::MarshallerFactory (Used to cre- ate marshallers for a specific version of the wire protocol)	2121
activemq::wireformat::openwire::marshal::v4::MarshallerFactory (Used to cre- ate marshallers for a specific version of the wire protocol)	2122
activemq::wireformat::openwire::marshal::v1::MarshallerFactory (Used to cre- ate marshallers for a specific version of the wire protocol)	2123

activemq::wireformat::openwire::marshal::v3::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2124
activemq::wireformat::openwire::marshal::v5::MarshallerFactory (Used to createmarshallers for a specific version of the wire protocol)	2125
decaf::lang::Math (The class Math (p.2126) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions)	2126
activemq::util::MemoryUsage	2141
activemq::commands::Message	2145
cms::Message (Root of all messages)	2163
activemq::commands::MessageAck	2188
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p.2194))	2194
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p.2198))	2198
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p.2202))	2202
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p.2206))	2206
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (Marshaling code for Open Wire Format for MessageAckMarshaller (p.2210))	2210
cms::MessageConsumer (A client uses a MessageConsumer (p.2214) to received messages from a destination)	2214
activemq::cmsutil::MessageCreator (Creates the user-defined message to be sent by the CmsTemplate (p.1041))	2218
activemq::commands::MessageDispatch	2219
activemq::core::MessageDispatchChannel	2224
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p.2231))	2231
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p.2235))	2235
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p.2239))	2239
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p.2243))	2243
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p.2247))	2247
activemq::commands::MessageDispatchNotification	2251
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p.2257))	2257
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p.2261))	2261

activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2265))	2265
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2269))	2269
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2273))	2273
cms::MessageEOFException (This exception must be thrown when an unexpected end of stream has been reached when a StreamMessage (p. 3081) or BytesMessage (p. 938) is being read)	2277
cms::MessageFormatException (This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type)	2278
activemq::commands::MessageId	2279
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2284))	2284
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2288))	2288
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2292))	2292
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2296))	2296
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2300))	2300
cms::MessageListener (A MessageListener (p. 2304) object is used to receive asynchronously delivered messages)	2304
activemq::wireformat::openwire::marshal::v2::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2305))	2305
activemq::wireformat::openwire::marshal::v4::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2310))	2310
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2315))	2315
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2320))	2320
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2325))	2325
cms::MessageNotReadableException (This exception must be thrown when a CMS client attempts to read a write-only message)	2330
cms::MessageNotWriteableException (This exception must be thrown when a CMS client attempts to write to a read-only message)	2331
cms::MessageProducer (A client uses a MessageProducer (p. 2332) object to send messages to a Destination (p. 1480))	2332
activemq::wireformat::openwire::utils::MessagePropertyInterceptor (Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties)	2339
activemq::commands::MessagePull	2346
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2351))	2351

activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2355))	2355
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2359))	2359
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2363))	2363
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2367))	2367
activemq::transport::mock::MockTransport (The MockTransport (p. 2371) defines a base level Transport (p. 3273) class that is intended to be used in place of an a regular protocol Transport (p. 3273) such as TCP)	2371
activemq::transport::mock::MockTransportFactory (Manufactures MockTransports , which are objects that read from input streams and write to output streams)	2382
decaf::util::concurrent::Mutex (Mutex (p. 2385) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java)	2385
decaf::util::concurrent::MutexHandle	2390
decaf::internal::util::concurrent::MutexImpl	2391
activemq::commands::NetworkBridgeFilter	2393
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2397))	2397
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2401))	2401
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2405))	2405
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2409))	2409
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2413))	2413
decaf::net::NoRouteToHostException	2417
decaf::security::NoSuchAlgorithmException	2420
decaf::lang::exceptions::NoSuchElementException	2423
decaf::security::NoSuchProviderException	2426
decaf::lang::exceptions::NullPointerException	2429
decaf::lang::Number (The abstract class Number (p. 2432) is the superclass of classes Byte (p. 840), Double (p. 1537), Float (p. 1647), Integer (p. 1762), Long (p. 2057), and Short (p. 2906))	2432
decaf::lang::exceptions::NumberFormatException	2435
cms::ObjectMessage (Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object)	2438
decaf::security_provider::unix::openssl::OpenSSLX500Principal (The OpenSSLX500Principal (p. 2439) wraps around an OpenSSL X509_NAME structure)	2439
decaf::security_provider::unix::openssl::OpenSSLX509Certificate	2442

activemq::wireformat::openwire::OpenWireFormat	2448
activemq::wireformat::openwire::OpenWireFormatFactory	2460
activemq::wireformat::openwire::OpenWireFormatNegotiator	2462
activemq::wireformat::openwire::OpenWireResponseBuilder	2466
activemq::wireformat::openwire::utils::OpenwireStringSupport	2468
decaf::io::OutputStream (Base interface for an output stream)	2470
activemq::commands::PartialCommand	2473
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2477))	2477
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2481))	2481
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2485))	2485
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2489))	2489
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2493))	2493
decaf::lang::Pointer< T, REFCOUNTER > (Decaf's implementation of a Smart Pointer (p. 2497) that is a template on a Type and is Thread Safe if the default Reference Counter is used)	2497
decaf::lang::PointerComparator< T, R > (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the Pointer (p. 2497) instance)	2504
activemq::cmsutil::PooledSession (A pooled session object that wraps around a del- egate session)	2505
decaf::util::concurrent::PooledThread	2518
decaf::util::concurrent::PooledThreadListener (Abstract Listener Interface for users of ThreadPool (p. 3175))	2520
decaf::net::PortUnreachableException	2522
activemq::util::PrimitiveList (List of primitives)	2525
activemq::util::PrimitiveMap (Map of named primitives)	2536
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller (This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire)	2546
activemq::util::PrimitiveValueNode::PrimitiveValue (Define a union type com- prised of the various types)	2551
activemq::util::PrimitiveValueConverter (Class controls the conversion of data con- tained in a PrimitiveValueNode (p. 2555) from one type to another)	2553
activemq::util::PrimitiveValueNode (Class that wraps around a single value of one of the many types)	2555
decaf::security::Principal (Base interface for a principal, which can represent an indi- vidual or organization)	2569
decaf::util::PriorityQueue< E > (An unbounded priority queue based on a binary heap algorithm)	2571
activemq::commands::ProducerAck	2579
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (Mar- shaling code for Open Wire Format for ProducerAckMarshaller (p. 2583))	2583

activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2587))	2587
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2591))	2591
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2595))	2595
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2599))	2599
activemq::cmsutil::ProducerCallback (Callback for sending a message to a CMS destination)	2603
activemq::cmsutil::CmsTemplate::ProducerExecutor	2604
activemq::commands::ProducerId	2606
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2611))	2611
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2615))	2615
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2619))	2619
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2623))	2623
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2627))	2627
activemq::commands::ProducerInfo	2631
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2636))	2636
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2640))	2640
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2644))	2644
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2648))	2648
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2652))	2652
activemq::state::ProducerState	2656
decaf::util::Properties (Java-like properties class for mapping string names to string values)	2657
decaf::util::logging::PropertiesChangeListener (Defines the interface that classes can use to listen for change events on Properties (p. 2657))	2666
decaf::net::ProtocolException	2667
decaf::security::PublicKey (A public key)	2670

decaf::util::Queue< E > (A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection)	2671
cms::Queue (An interface encapsulating a provider-specific queue name)	2674
cms::QueueBrowser (This class implements in interface for browsing the messages in a Queue (p. 2674) without removing them)	2675
decaf::util::Random (Random (p. 2677) Value Generator which is used to generate a stream of pseudorandom numbers)	2677
activemq::transport::inactivity::ReadChecker (Runnable class that is used by the {)	2682
decaf::io::Reader	2683
decaf::nio::ReadOnlyBufferException	2685
decaf::util::concurrent::locks::ReadWriteLock (A ReadWriteLock (p. 2688) maintains a pair of associated locks, one for read-only operations and one for writing)	2688
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2690
decaf::util::concurrent::locks::ReentrantLock (A reentrant mutual exclusion Lock (p. 2017) with extended capabilities)	2692
decaf::util::concurrent::RejectedExecutionException	2699
decaf::util::concurrent::RejectedExecutionHandler (A handler for tasks that cannot be executed by a ThreadPoolExecutor (p. ??))	2702
activemq::commands::RemoveInfo	2703
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2707))	2707
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2711))	2711
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2715))	2715
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2719))	2719
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2723))	2723
activemq::commands::RemoveSubscriptionInfo	2727
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2732))	2732
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2736))	2736
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2740))	2740
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2744))	2744
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2748))	2748
activemq::commands::ReplayCommand	2752

activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2756))	2756
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2760))	2760
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2764))	2764
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2768))	2768
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2772))	2772
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	2776
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	2777
activemq::cmsutil::ResourceLifecycleManager (Manages the lifecycle of a set of CMS resources)	2778
activemq::commands::Response	2781
activemq::transport::mock::ResponseBuilder (Interface for all Protocols to imple- ment that defines the behavior of the Broker in response to messages of that protocol)	2785
activemq::transport::correlator::ResponseCorrelator (This type of transport filter is responsible for correlating asynchronous responses with requests)	2787
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2791))	2791
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2796))	2796
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2801))	2801
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2806))	2806
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2811))	2811
decaf::lang::Runnable (Interface for a runnable object - defines a task that can be run by a thread)	2816
decaf::lang::Runtime	2817
decaf::lang::exceptions::RuntimeException	2819
decaf::security_provider::SecurityProvider	2822
decaf::security_provider::SecurityProviderMap (Lookup Map for Connector Fac- tories)	2823
decaf::security_provider::SecurityProviderRegistrar (Registers the passed in provider into the provider map, this class can manage the lifetime of the regis- tered provider (default behaviour))	2825
decaf::util::concurrent::Semaphore (A counting semaphore)	2827
activemq::cmsutil::CmsTemplate::SendExecutor	2836
decaf::net::ServerSocket (A server socket class (for testing purposes))	2837
cms::Session (A Session (p. 2839) object is a single-threaded context for producing and consuming messages)	2839
activemq::cmsutil::SessionCallback (Callback for executing any number of opera- tions on a provided CMS Session)	2852
activemq::commands::SessionId	2853

activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2857))	2857
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2861))	2861
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2865))	2865
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2869))	2869
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2873))	2873
activemq::commands::SessionInfo	2877
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2881))	2881
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2885))	2885
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2889))	2889
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2893))	2893
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2897))	2897
activemq::cmsutil::SessionPool (A pool of CMS sessions from the same connection and with the same acknowledge mode)	2901
activemq::state::SessionState	2903
decaf::util::Set< E > (A collection that contains no duplicate elements)	2905
decaf::lang::Short	2906
decaf::internal::nio::ShortArrayBuffer	2915
decaf::nio::ShortBuffer (This class defines four categories of operations upon short buffers:)	2923
activemq::commands::ShutdownInfo	2934
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2937))	2937
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2941))	2941
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2945))	2945
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2949))	2949
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2953))	2953
decaf::security::SignatureException	2957
decaf::util::logging::SimpleFormatter (Print a brief summary of the LogRecord (p. 2050) in a human readable format)	2960
decaf::util::logging::SimpleLogger	2962
decaf::net::Socket	2964
decaf::net::SocketError (Static utility class to simplify handling of error codes for socket operations)	2971
decaf::net::SocketException (Exception for errors when manipulating sockets)	2972

decaf::net::SocketFactory (Socket (p. 2964) Factory implementation for use in Creating Sockets)	2975
decaf::net::SocketInputStream (Input stream for performing reads on a socket) . .	2977
decaf::net::SocketOutputStream (Output stream for performing write operations on a socket)	2984
decaf::net::SocketTimeoutException	2990
activemq::commands::BrokerError::StackTraceElement	2993
decaf::internal::io::StandardErrorOutputStream (Wrapper Around the Standard error Output facility on the current platform)	2994
decaf::internal::io::StandardInputStream	3000
decaf::internal::io::StandardOutputStream	3008
cms::Startable (Interface for a class that implements the start method)	3014
decaf::lang::STATIC_CAST_TOKEN	3015
activemq::core::ActiveMQConstants::StaticInitializer	3016
decaf::util::StlList< E > (List (p. 1984) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type) . .	3017
decaf::util::StlMap< K, V, COMPARATOR > (Map (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map)	3030
decaf::util::StlQueue< T > (The Queue (p. 2671) class accepts messages with an psuh(m) command where m is the message to be queued)	3043
decaf::util::StlSet< E > (Set (p. 2905) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set)	3051
activemq::wireformat::stomp::StompCommandConstants	3057
activemq::wireformat::stomp::StompFrame (A Stomp-level message frame that encloses all messages to and from the broker)	3061
activemq::wireformat::stomp::StompHelper (Utility Methods used when marshaling to and from StompFrame's)	3066
activemq::wireformat::stomp::StompWireFormat	3071
activemq::wireformat::stomp::StompWireFormatFactory (Factory used to create the Stomp Wire Format instance)	3075
cms::Stoppable (Interface for a class that implements the stop method)	3076
decaf::util::logging::StreamHandler	3077
cms::StreamMessage (Interface for a StreamMessage (p. 3081))	3081
decaf::util::StringTokenizer	3094
activemq::commands::SubscriptionInfo	3097
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3102))	3102
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3106))	3106
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3110))	3110
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3114))	3114
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 3118))	3118
decaf::util::concurrent::Synchronizable (The interface for all synchronizable objects (that is, objects that can be locked and unlocked))	3122

decaf::internal::util::concurrent::SynchronizableImpl (A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance)	3133
activemq::core::Synchronization (Transacted Object Synchronization (p. 3137), used to sync the events of a Transaction with the items in the Transaction)	3137
decaf::util::concurrent::SynchronousQueue< E > (A BlockingQueue blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa)	3138
decaf::lang::System	3145
activemq::threads::Task (Represents a unit of work that requires one or more iterations to complete)	3148
decaf::util::concurrent::TaskListener	3149
activemq::threads::TaskRunner	3150
decaf::net::TcpSocket (Platform-independent implementation of the socket interface)	3152
activemq::transport::tcp::TcpTransport (Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an IOTransport (p. 1823))	3160
activemq::transport::tcp::TcpTransportFactory (Factory Responsible for creating the TcpTransport (p. 3160))	3163
cms::TemporaryQueue (Defines a Temporary Queue (p. 2674) based Destination (p. 1480))	3166
cms::TemporaryTopic (Defines a Temporary Topic (p. 3215) based Destination (p. 1480))	3168
cms::TextMessage (Interface for a text message)	3170
decaf::util::concurrent::ThreadFactory (Public interface ThreadFactory (p. 3172))	3172
decaf::lang::ThreadGroup	3174
decaf::util::concurrent::ThreadPool (Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks)	3175
decaf::lang::Throwable (This class represents an error that has occurred)	3181
decaf::util::concurrent::TimeoutException	3185
decaf::util::Timer (A facility for threads to schedule tasks for future execution in a background thread)	3188
decaf::util::TimerTask (A Base class for a task object that can be scheduled for one-time or repeated execution by a Timer (p. 3188))	3199
decaf::internal::util::TimerTaskHeap (A Binary Heap implemented specifically for the Timer class in Decaf Util)	3202
decaf::util::concurrent::TimeUnit (A TimeUnit (p. 3205) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units)	3205
cms::Topic (An interface encapsulating a provider-specific topic name)	3215
activemq::state::Tracked	3216
activemq::commands::TransactionId	3217
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3220))	3220
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3224))	3224
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3228))	3228

activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3232))	3232
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3236))	3236
activemq::commands::TransactionInfo	3240
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3245))	3245
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3249))	3249
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3253))	3253
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3257))	3257
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3261))	3261
activemq::state::TransactionState	3265
decaf::internal::util::concurrent::Transferer< E > (Shared internal API for dual stacks and queues)	3267
decaf::internal::util::concurrent::TransferQueue< E > (This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers)	3268
decaf::internal::util::concurrent::TransferStack< E >	3271
activemq::transport::Transport (Interface for a transport layer for command objects)	3273
activemq::transport::TransportFactory (Defines the interface for Factories that create Transports or TransportFilters)	3279
activemq::transport::TransportFilter (A filter on the transport layer)	3281
activemq::transport::TransportListener (A listener of asynchronous exceptions from a command transport object)	3289
activemq::transport::TransportRegistry (Registry of all Transport (p. 3273) Factories that are available to the client at runtime)	3291
decaf::net::UnknownHostException	3294
decaf::net::UnknownServiceException	3297
decaf::io::UnsupportedEncodingException (Thrown when the the Character Encoding is not supported)	3300
decaf::lang::exceptions::UnsupportedOperationException	3303
cms::UnsupportedOperationException (This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use)	3306
decaf::net::URI (This class represents an instance of a URI (p. 3308) as defined by RFC 2396)	3308
decaf::internal::net::URIEncoderDecoder	3319
decaf::internal::net::URIHelper (Helper class used by the URI classes in encoding and decoding of URI's)	3322
activemq::transport::failover::URIPool	3329
activemq::util::URISupport	3332
decaf::net::URISyntaxException	3335

decaf::internal::net::URIType (Basic type object that holds data that composes a given URI)	3339
decaf::net::URL (Class URL (p. 3347) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web)	3347
decaf::net::URLDecoder	3349
decaf::net::URLEncoder	3350
activemq::util::Usage	3351
decaf::io::UTFDataFormatException (Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered)	3353
decaf::util::UUID (A class that represents an immutable universally unique identifier (UUID (p. 3356)))	3356
activemq::wireformat::WireFormat (Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams) .	3363
activemq::wireformat::WireFormatFactory (The WireFormatFactory (p. 3367) is the interface that all WireFormatFactory (p. 3367) classes must extend) .	3367
activemq::commands::WireFormatInfo	3369
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3379))	3379
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3383))	3383
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3387))	3387
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3391))	3391
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p. 3395))	3395
activemq::wireformat::WireFormatNegotiator (Defines a WireFormatNegotiator (p. 3399) which allows a WireFormat (p. 3363) to)	3399
activemq::wireformat::WireFormatRegistry (Registry of all WireFormat (p. 3363) Factories that are available to the client at runtime)	3400
activemq::transport::inactivity::WriteChecker (Runnable class used by the { }) .	3403
decaf::io::Writer	3404
decaf::security::auth::x500::X500Principal	3406
decaf::security::cert::X509Certificate (Base interface for all identity certificates) .	3407
activemq::commands::XATransactionId	3410
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3415))	3415
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3419))	3419
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3423))	3423
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 3427))	3427

activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	
(Marshaling code for Open Wire Format for XATransactionIdMarshaller	
(p. 3431)	3431

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/	CachedConsumer.h	3435
src/main/activemq/cmsutil/	CachedProducer.h	3436
src/main/activemq/cmsutil/	CmsAccessor.h	3437
src/main/activemq/cmsutil/	CmsDestinationAccessor.h	3438
src/main/activemq/cmsutil/	CmsTemplate.h	3439
src/main/activemq/cmsutil/	DestinationResolver.h	3440
src/main/activemq/cmsutil/	DynamicDestinationResolver.h	3441
src/main/activemq/cmsutil/	MessageCreator.h	3442
src/main/activemq/cmsutil/	PooledSession.h	3443
src/main/activemq/cmsutil/	ProducerCallback.h	3444
src/main/activemq/cmsutil/	ResourceLifecycleManager.h	3445
src/main/activemq/cmsutil/	SessionCallback.h	3446
src/main/activemq/cmsutil/	SessionPool.h	3447
src/main/activemq/commands/	ActiveMQBlobMessage.h	3448
src/main/activemq/commands/	ActiveMQBytesMessage.h	3449
src/main/activemq/commands/	ActiveMQDestination.h	3450
src/main/activemq/commands/	ActiveMQMapMessage.h	3451
src/main/activemq/commands/	ActiveMQMessage.h	3452
src/main/activemq/commands/	ActiveMQMessageTemplate.h	3453
src/main/activemq/commands/	ActiveMQObjectMessage.h	3454
src/main/activemq/commands/	ActiveMQQueue.h	3455
src/main/activemq/commands/	ActiveMQStreamMessage.h	3456
src/main/activemq/commands/	ActiveMQTempDestination.h	3457
src/main/activemq/commands/	ActiveMQTempQueue.h	3458
src/main/activemq/commands/	ActiveMQTempTopic.h	3459
src/main/activemq/commands/	ActiveMQTextMessage.h	3460
src/main/activemq/commands/	ActiveMQTopic.h	3461
src/main/activemq/commands/	BaseCommand.h	3462
src/main/activemq/commands/	BaseDataStructure.h	3463
src/main/activemq/commands/	BooleanExpression.h	3464
src/main/activemq/commands/	BrokerError.h	3465
src/main/activemq/commands/	BrokerId.h	3466
src/main/activemq/commands/	BrokerInfo.h	3467

src/main/activemq/commands/	Command.h	3468
src/main/activemq/commands/	ConnectionControl.h	3469
src/main/activemq/commands/	ConnectionError.h	3470
src/main/activemq/commands/	ConnectionId.h	3471
src/main/activemq/commands/	ConnectionInfo.h	3472
src/main/activemq/commands/	ConsumerControl.h	3473
src/main/activemq/commands/	ConsumerId.h	3474
src/main/activemq/commands/	ConsumerInfo.h	3475
src/main/activemq/commands/	ControlCommand.h	3476
src/main/activemq/commands/	DataArrayResponse.h	3477
src/main/activemq/commands/	DataResponse.h	3478
src/main/activemq/commands/	DataStructure.h	3479
src/main/activemq/commands/	DestinationInfo.h	3480
src/main/activemq/commands/	DiscoveryEvent.h	3481
src/main/activemq/commands/	ExceptionResponse.h	3482
src/main/activemq/commands/	FlushCommand.h	3483
src/main/activemq/commands/	IntegerResponse.h	3484
src/main/activemq/commands/	JournalQueueAck.h	3485
src/main/activemq/commands/	JournalTopicAck.h	3486
src/main/activemq/commands/	JournalTrace.h	3487
src/main/activemq/commands/	JournalTransaction.h	3488
src/main/activemq/commands/	KeepAliveInfo.h	3489
src/main/activemq/commands/	LastPartialCommand.h	3490
src/main/activemq/commands/	LocalTransactionId.h	3491
src/main/activemq/commands/	Message.h	3492
src/main/activemq/commands/	MessageAck.h	3494
src/main/activemq/commands/	MessageDispatch.h	3495
src/main/activemq/commands/	MessageDispatchNotification.h	3496
src/main/activemq/commands/	MessageId.h	3497
src/main/activemq/commands/	MessagePull.h	3498
src/main/activemq/commands/	NetworkBridgeFilter.h	3499
src/main/activemq/commands/	PartialCommand.h	3500
src/main/activemq/commands/	ProducerAck.h	3501
src/main/activemq/commands/	ProducerId.h	3502
src/main/activemq/commands/	ProducerInfo.h	3503
src/main/activemq/commands/	RemoveInfo.h	3504
src/main/activemq/commands/	RemoveSubscriptionInfo.h	3505
src/main/activemq/commands/	ReplayCommand.h	3506
src/main/activemq/commands/	Response.h	3507
src/main/activemq/commands/	SessionId.h	3508
src/main/activemq/commands/	SessionInfo.h	3509
src/main/activemq/commands/	ShutdownInfo.h	3510
src/main/activemq/commands/	SubscriptionInfo.h	3511
src/main/activemq/commands/	TransactionId.h	3512
src/main/activemq/commands/	TransactionInfo.h	3513
src/main/activemq/commands/	WireFormatInfo.h	3514
src/main/activemq/commands/	XATransactionId.h	3515
src/main/activemq/core/	ActiveMQAckHandler.h	3516
src/main/activemq/core/	ActiveMQConnection.h	3517
src/main/activemq/core/	ActiveMQConnectionFactory.h	3518
src/main/activemq/core/	ActiveMQConnectionMetaData.h	3519
src/main/activemq/core/	ActiveMQConnectionSupport.h	3520
src/main/activemq/core/	ActiveMQConstants.h	3521
src/main/activemq/core/	ActiveMQConsumer.h	3522

src/main/activemq/core/ActiveMQProducer.h	3523
src/main/activemq/core/ActiveMQSession.h	3524
src/main/activemq/core/ActiveMQSessionExecutor.h	3525
src/main/activemq/core/ActiveMQTransactionContext.h	3526
src/main/activemq/core/DispatchData.h	3527
src/main/activemq/core/Dispatcher.h	3528
src/main/activemq/core/MessageDispatchChannel.h	3529
src/main/activemq/core/Synchronization.h	3530
src/main/activemq/exceptions/ActiveMQException.h	3531
src/main/activemq/exceptions/BrokerException.h	3532
src/main/activemq/exceptions/ExceptionDefines.h	3533
src/main/activemq/io/LoggingInputStream.h	3539
src/main/activemq/io/LoggingOutputStream.h	3540
src/main/activemq/library/ActiveMQCPP.h	3541
src/main/activemq/state/CommandVisitor.h	3542
src/main/activemq/state/CommandVisitorAdapter.h	3543
src/main/activemq/state/ConnectionState.h	3545
src/main/activemq/state/ConnectionStateTracker.h	3546
src/main/activemq/state/ConsumerState.h	3547
src/main/activemq/state/ProducerState.h	3548
src/main/activemq/state/SessionState.h	3549
src/main/activemq/state/Tracked.h	3550
src/main/activemq/state/TransactionState.h	3551
src/main/activemq/threads/CompositeTask.h	3552
src/main/activemq/threads/CompositeTaskRunner.h	3553
src/main/activemq/threads/DedicatedTaskRunner.h	3554
src/main/activemq/threads/Task.h	3555
src/main/activemq/threads/TaskRunner.h	3556
src/main/activemq/transport/AbstractTransportFactory.h	3557
src/main/activemq/transport/CompositeTransport.h	3558
src/main/activemq/transport/DefaultTransportListener.h	3561
src/main/activemq/transport/IOTransport.h	3572
src/main/activemq/transport/Transport.h	3580
src/main/activemq/transport/TransportFactory.h	3581
src/main/activemq/transport/TransportFilter.h	3582
src/main/activemq/transport/TransportListener.h	3583
src/main/activemq/transport/TransportRegistry.h	3584
src/main/activemq/transport/correlator/FutureResponse.h	3559
src/main/activemq/transport/correlator/ResponseCorrelator.h	3560
src/main/activemq/transport/failover/BackupTransport.h	3562
src/main/activemq/transport/failover/BackupTransportPool.h	3563
src/main/activemq/transport/failover/CloseTransportsTask.h	3564
src/main/activemq/transport/failover/FailoverTransport.h	3565
src/main/activemq/transport/failover/FailoverTransportFactory.h	3566
src/main/activemq/transport/failover/FailoverTransportListener.h	3567
src/main/activemq/transport/failover/URIPool.h	3568
src/main/activemq/transport/inactivity/InactivityMonitor.h	3569
src/main/activemq/transport/inactivity/ReadChecker.h	3570
src/main/activemq/transport/inactivity/WriteChecker.h	3571
src/main/activemq/transport/logging/LoggingTransport.h	3573
src/main/activemq/transport/mock/InternalCommandListener.h	3574
src/main/activemq/transport/mock/MockTransport.h	3575
src/main/activemq/transport/mock/MockTransportFactory.h	3576
src/main/activemq/transport/mock/ResponseBuilder.h	3577

src/main/activemq/transport/tcp/ TcpTransport.h	3578
src/main/activemq/transport/tcp/ TcpTransportFactory.h	3579
src/main/activemq/util/ ActiveMQProperties.h	3585
src/main/activemq/util/ CMSExceptionSupport.h	3586
src/main/activemq/util/ CompositeData.h	3588
src/main/activemq/util/ Config.h	3589
src/main/activemq/util/ LongSequenceGenerator.h	3592
src/main/activemq/util/ MemoryUsage.h	3593
src/main/activemq/util/ PrimitiveList.h	3594
src/main/activemq/util/ PrimitiveMap.h	3595
src/main/activemq/util/ PrimitiveValueConverter.h	3596
src/main/activemq/util/ PrimitiveValueNode.h	3597
src/main/activemq/util/ URISupport.h	3598
src/main/activemq/util/ Usage.h	3599
src/main/activemq/wireformat/ MarshalAware.h	3600
src/main/activemq/wireformat/ WireFormat.h	3927
src/main/activemq/wireformat/ WireFormatFactory.h	3928
src/main/activemq/wireformat/ WireFormatNegotiator.h	3929
src/main/activemq/wireformat/ WireFormatRegistry.h	3930
src/main/activemq/wireformat/openwire/ OpenWireFormat.h	3914
src/main/activemq/wireformat/openwire/ OpenWireFormatFactory.h	3915
src/main/activemq/wireformat/openwire/ OpenWireFormatNegotiator.h	3916
src/main/activemq/wireformat/openwire/ OpenWireResponseBuilder.h	3917
src/main/activemq/wireformat/openwire/marshal/ BaseDataStreamMarshaller.h	3601
src/main/activemq/wireformat/openwire/marshal/ DataStreamMarshaller.h	3602
src/main/activemq/wireformat/openwire/marshal/ PrimitiveTypesMarshaller.h	3603
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQBlobMessageMarshaller.h	3604
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQBytesMessageMarshaller.h	3609
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQDestinationMarshaller.h	3614
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQMapMessageMarshaller.h	3619
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQMessageMarshaller.h	3624
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQObjectMessageMarshaller.h	3629
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQQueueMarshaller.h	3634
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQStreamMessageMarshaller.h	3639
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempDestinationMarshaller.h	3644
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempQueueMarshaller.h	3649
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempTopicMarshaller.h	3654
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTextMessageMarshaller.h	3659
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTopicMarshaller.h	3664
src/main/activemq/wireformat/openwire/marshal/v1/ BaseCommandMarshaller.h	3669
src/main/activemq/wireformat/openwire/marshal/v1/ BrokerIdMarshaller.h	3674
src/main/activemq/wireformat/openwire/marshal/v1/ BrokerInfoMarshaller.h	3679

src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionControlMarshaller.h	
3684		
src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionErrorMarshaller.h	
3689		
src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionIdMarshaller.h	3694
src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionInfoMarshaller.h	3699
src/main/activemq/wireformat/openwire/marshal/v1/	ConsumerControlMarshaller.h	
3704		
src/main/activemq/wireformat/openwire/marshal/v1/	ConsumerIdMarshaller.h	3709
src/main/activemq/wireformat/openwire/marshal/v1/	ConsumerInfoMarshaller.h	3714
src/main/activemq/wireformat/openwire/marshal/v1/	ControlCommandMarshaller.h	
3719		
src/main/activemq/wireformat/openwire/marshal/v1/	DataArrayResponseMarshaller.h	
3724		
src/main/activemq/wireformat/openwire/marshal/v1/	DataResponseMarshaller.h	3729
src/main/activemq/wireformat/openwire/marshal/v1/	DestinationInfoMarshaller.h	3734
src/main/activemq/wireformat/openwire/marshal/v1/	DiscoveryEventMarshaller.h	3739
src/main/activemq/wireformat/openwire/marshal/v1/	ExceptionResponseMarshaller.h	
3744		
src/main/activemq/wireformat/openwire/marshal/v1/	FlushCommandMarshaller.h	3749
src/main/activemq/wireformat/openwire/marshal/v1/	IntegerResponseMarshaller.h	
3754		
src/main/activemq/wireformat/openwire/marshal/v1/	JournalQueueAckMarshaller.h	
3759		
src/main/activemq/wireformat/openwire/marshal/v1/	JournalTopicAckMarshaller.h	
3764		
src/main/activemq/wireformat/openwire/marshal/v1/	JournalTraceMarshaller.h	3769
src/main/activemq/wireformat/openwire/marshal/v1/	JournalTransactionMarshaller.h	
3774		
src/main/activemq/wireformat/openwire/marshal/v1/	KeepAliveInfoMarshaller.h	3779
src/main/activemq/wireformat/openwire/marshal/v1/	LastPartialCommandMarshaller.h	
3784		
src/main/activemq/wireformat/openwire/marshal/v1/	LocalTransactionIdMarshaller.h	
3789		
src/main/activemq/wireformat/openwire/marshal/v1/	MarshallerFactory.h	3794
src/main/activemq/wireformat/openwire/marshal/v1/	MessageAckMarshaller.h	3799
src/main/activemq/wireformat/openwire/marshal/v1/	MessageDispatchMarshaller.h	
3804		
src/main/activemq/wireformat/openwire/marshal/v1/	MessageDispatchNotificationMarshaller.h	
3809		
src/main/activemq/wireformat/openwire/marshal/v1/	MessageIdMarshaller.h	3814
src/main/activemq/wireformat/openwire/marshal/v1/	MessageMarshaller.h	3819
src/main/activemq/wireformat/openwire/marshal/v1/	MessagePullMarshaller.h	3824
src/main/activemq/wireformat/openwire/marshal/v1/	NetworkBridgeFilterMarshaller.h	
3829		
src/main/activemq/wireformat/openwire/marshal/v1/	PartialCommandMarshaller.h	
3834		
src/main/activemq/wireformat/openwire/marshal/v1/	ProducerAckMarshaller.h	3839
src/main/activemq/wireformat/openwire/marshal/v1/	ProducerIdMarshaller.h	3844
src/main/activemq/wireformat/openwire/marshal/v1/	ProducerInfoMarshaller.h	3849
src/main/activemq/wireformat/openwire/marshal/v1/	RemoveInfoMarshaller.h	3854
src/main/activemq/wireformat/openwire/marshal/v1/	RemoveSubscriptionInfoMarshaller.h	
3859		

src/main/activemq/wireformat/openwire/marshal/v1/ ReplayCommandMarshaller.h	
3864	
src/main/activemq/wireformat/openwire/marshal/v1/ ResponseMarshaller.h	3869
src/main/activemq/wireformat/openwire/marshal/v1/ SessionIdMarshaller.h	3874
src/main/activemq/wireformat/openwire/marshal/v1/ SessionInfoMarshaller.h . . .	3879
src/main/activemq/wireformat/openwire/marshal/v1/ ShutdownInfoMarshaller.h .	3884
src/main/activemq/wireformat/openwire/marshal/v1/ SubscriptionInfoMarshaller.h	
3889	
src/main/activemq/wireformat/openwire/marshal/v1/ TransactionIdMarshaller.h .	3894
src/main/activemq/wireformat/openwire/marshal/v1/ TransactionInfoMarshaller.h	3899
src/main/activemq/wireformat/openwire/marshal/v1/ WireFormatInfoMarshaller.h	3904
src/main/activemq/wireformat/openwire/marshal/v1/ XATransactionIdMarshaller.h	
3909	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQBlobMessageMarshaller.h	
3605	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQBytesMessageMarshaller.h	
3610	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQDestinationMarshaller.h	
3615	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQMapMessageMarshaller.h	
3620	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQMessageMarshaller.h	
3625	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQObjectMessageMarshaller.h	
3630	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQQueueMarshaller.h	
3635	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQStreamMessageMarshaller.h	
3640	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTempDestinationMarshaller.h	
3645	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTempQueueMarshaller.h	
3650	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTempTopicMarshaller.h	
3655	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTextMessageMarshaller.h	
3660	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTopicMarshaller.h	3665
src/main/activemq/wireformat/openwire/marshal/v2/ BaseCommandMarshaller.h	3670
src/main/activemq/wireformat/openwire/marshal/v2/ BrokerIdMarshaller.h	3675
src/main/activemq/wireformat/openwire/marshal/v2/ BrokerInfoMarshaller.h . . .	3680
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionControlMarshaller.h	
3685	
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionErrorMarshaller.h	
3690	
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionIdMarshaller.h .	3695
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionInfoMarshaller.h	3700
src/main/activemq/wireformat/openwire/marshal/v2/ ConsumerControlMarshaller.h	
3705	
src/main/activemq/wireformat/openwire/marshal/v2/ ConsumerIdMarshaller.h . .	3710
src/main/activemq/wireformat/openwire/marshal/v2/ ConsumerInfoMarshaller.h .	3715
src/main/activemq/wireformat/openwire/marshal/v2/ ControlCommandMarshaller.h	
3720	

src/main/activemq/wireformat/openwire/marshall/v2/ DataArrayResponseMarshaller.h	
3725	
src/main/activemq/wireformat/openwire/marshall/v2/ DataResponseMarshaller.h	3730
src/main/activemq/wireformat/openwire/marshall/v2/ DestinationInfoMarshaller.h	3735
src/main/activemq/wireformat/openwire/marshall/v2/ DiscoveryEventMarshaller.h	3740
src/main/activemq/wireformat/openwire/marshall/v2/ ExceptionResponseMarshaller.h	
3745	
src/main/activemq/wireformat/openwire/marshall/v2/ FlushCommandMarshaller.h	3750
src/main/activemq/wireformat/openwire/marshall/v2/ IntegerResponseMarshaller.h	
3755	
src/main/activemq/wireformat/openwire/marshall/v2/ JournalQueueAckMarshaller.h	
3760	
src/main/activemq/wireformat/openwire/marshall/v2/ JournalTopicAckMarshaller.h	
3765	
src/main/activemq/wireformat/openwire/marshall/v2/ JournalTraceMarshaller.h	3770
src/main/activemq/wireformat/openwire/marshall/v2/ JournalTransactionMarshaller.h	
3775	
src/main/activemq/wireformat/openwire/marshall/v2/ KeepAliveInfoMarshaller.h	3780
src/main/activemq/wireformat/openwire/marshall/v2/ LastPartialCommandMarshaller.h	
3785	
src/main/activemq/wireformat/openwire/marshall/v2/ LocalTransactionIdMarshaller.h	
3790	
src/main/activemq/wireformat/openwire/marshall/v2/ MarshallerFactory.h	3795
src/main/activemq/wireformat/openwire/marshall/v2/ MessageAckMarshaller.h	3800
src/main/activemq/wireformat/openwire/marshall/v2/ MessageDispatchMarshaller.h	
3805	
src/main/activemq/wireformat/openwire/marshall/v2/ MessageDispatchNotificationMarshaller.h	
3810	
src/main/activemq/wireformat/openwire/marshall/v2/ MessageIdMarshaller.h	3815
src/main/activemq/wireformat/openwire/marshall/v2/ MessageMarshaller.h	3820
src/main/activemq/wireformat/openwire/marshall/v2/ MessagePullMarshaller.h	3825
src/main/activemq/wireformat/openwire/marshall/v2/ NetworkBridgeFilterMarshaller.h	
3830	
src/main/activemq/wireformat/openwire/marshall/v2/ PartialCommandMarshaller.h	
3835	
src/main/activemq/wireformat/openwire/marshall/v2/ ProducerAckMarshaller.h	3840
src/main/activemq/wireformat/openwire/marshall/v2/ ProducerIdMarshaller.h	3845
src/main/activemq/wireformat/openwire/marshall/v2/ ProducerInfoMarshaller.h	3850
src/main/activemq/wireformat/openwire/marshall/v2/ RemoveInfoMarshaller.h	3855
src/main/activemq/wireformat/openwire/marshall/v2/ RemoveSubscriptionInfoMarshaller.h	
3860	
src/main/activemq/wireformat/openwire/marshall/v2/ ReplayCommandMarshaller.h	
3865	
src/main/activemq/wireformat/openwire/marshall/v2/ ResponseMarshaller.h	3870
src/main/activemq/wireformat/openwire/marshall/v2/ SessionIdMarshaller.h	3875
src/main/activemq/wireformat/openwire/marshall/v2/ SessionInfoMarshaller.h	3880
src/main/activemq/wireformat/openwire/marshall/v2/ ShutdownInfoMarshaller.h	3885
src/main/activemq/wireformat/openwire/marshall/v2/ SubscriptionInfoMarshaller.h	
3890	
src/main/activemq/wireformat/openwire/marshall/v2/ TransactionIdMarshaller.h	3895
src/main/activemq/wireformat/openwire/marshall/v2/ TransactionInfoMarshaller.h	3900
src/main/activemq/wireformat/openwire/marshall/v2/ WireFormatInfoMarshaller.h	3905
src/main/activemq/wireformat/openwire/marshall/v2/ XATransactionIdMarshaller.h	
3910	

src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h	
3606	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h	
3611	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
3616	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
3621	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
3626	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h	
3631	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h	
3636	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h	
3641	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h	
3646	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h	
3651	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h	
3656	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h	
3661	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	3666
src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	3671
src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	3676
src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h	3681
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
3686	
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
3691	
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	3696
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h	3701
src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h	
3706	
src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	3711
src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	3716
src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
3721	
src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
3726	
src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h	3731
src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h	3736
src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h	3741
src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h	
3746	
src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	3751
src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
3756	
src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
3761	
src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
3766	

src/main/activemq/wireformat/openwire/marshal/v3/ JournalTraceMarshaller.h . . .	3771
src/main/activemq/wireformat/openwire/marshal/v3/ JournalTransactionMarshaller.h	3776
src/main/activemq/wireformat/openwire/marshal/v3/ KeepAliveInfoMarshaller.h . .	3781
src/main/activemq/wireformat/openwire/marshal/v3/ LastPartialCommandMarshaller.h	3786
src/main/activemq/wireformat/openwire/marshal/v3/ LocalTransactionIdMarshaller.h	3791
src/main/activemq/wireformat/openwire/marshal/v3/ MarshallerFactory.h	3796
src/main/activemq/wireformat/openwire/marshal/v3/ MessageAckMarshaller.h . . .	3801
src/main/activemq/wireformat/openwire/marshal/v3/ MessageDispatchMarshaller.h	3806
src/main/activemq/wireformat/openwire/marshal/v3/ MessageDispatchNotificationMarshaller.h	3811
src/main/activemq/wireformat/openwire/marshal/v3/ MessageIdMarshaller.h . . .	3816
src/main/activemq/wireformat/openwire/marshal/v3/ MessageMarshaller.h	3821
src/main/activemq/wireformat/openwire/marshal/v3/ MessagePullMarshaller.h . .	3826
src/main/activemq/wireformat/openwire/marshal/v3/ NetworkBridgeFilterMarshaller.h	3831
src/main/activemq/wireformat/openwire/marshal/v3/ PartialCommandMarshaller.h	3836
src/main/activemq/wireformat/openwire/marshal/v3/ ProducerAckMarshaller.h . .	3841
src/main/activemq/wireformat/openwire/marshal/v3/ ProducerIdMarshaller.h . . .	3846
src/main/activemq/wireformat/openwire/marshal/v3/ ProducerInfoMarshaller.h . .	3851
src/main/activemq/wireformat/openwire/marshal/v3/ RemoveInfoMarshaller.h . .	3856
src/main/activemq/wireformat/openwire/marshal/v3/ RemoveSubscriptionInfoMarshaller.h	3861
src/main/activemq/wireformat/openwire/marshal/v3/ ReplayCommandMarshaller.h	3866
src/main/activemq/wireformat/openwire/marshal/v3/ ResponseMarshaller.h	3871
src/main/activemq/wireformat/openwire/marshal/v3/ SessionIdMarshaller.h	3876
src/main/activemq/wireformat/openwire/marshal/v3/ SessionInfoMarshaller.h . . .	3881
src/main/activemq/wireformat/openwire/marshal/v3/ ShutdownInfoMarshaller.h .	3886
src/main/activemq/wireformat/openwire/marshal/v3/ SubscriptionInfoMarshaller.h	3891
src/main/activemq/wireformat/openwire/marshal/v3/ TransactionIdMarshaller.h .	3896
src/main/activemq/wireformat/openwire/marshal/v3/ TransactionInfoMarshaller.h	3901
src/main/activemq/wireformat/openwire/marshal/v3/ WireFormatInfoMarshaller.h	3906
src/main/activemq/wireformat/openwire/marshal/v3/ XATransactionIdMarshaller.h	3911
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQBlobMessageMarshaller.h	3607
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQBytesMessageMarshaller.h	3612
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQDestinationMarshaller.h	3617
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQMapMessageMarshaller.h	3622
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQMessageMarshaller.h	3627
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQObjectMessageMarshaller.h	3632
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQQueueMarshaller.h	3637

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h	
3642	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h	
3647	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h	
3652	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h	
3657	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h	
3662	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	3667
src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	3672
src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	3677
src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h	3682
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
3687	
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
3692	
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	3697
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h	3702
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h	
3707	
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	3712
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	3717
src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
3722	
src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
3727	
src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h	3732
src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h	3737
src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h	3742
src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h	
3747	
src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	3752
src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
3757	
src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
3762	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
3767	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	3772
src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
3777	
src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	3782
src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
3787	
src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
3792	
src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	3797
src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h	3802
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
3807	
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
3812	

src/main/activemq/wireformat/openwire/marshall/v4/MessageIdMarshaller.h . . .	3817
src/main/activemq/wireformat/openwire/marshall/v4/MessageMarshaller.h	3822
src/main/activemq/wireformat/openwire/marshall/v4/MessagePullMarshaller.h . .	3827
src/main/activemq/wireformat/openwire/marshall/v4/NetworkBridgeFilterMarshaller.h	
3832	
src/main/activemq/wireformat/openwire/marshall/v4/PartialCommandMarshaller.h	
3837	
src/main/activemq/wireformat/openwire/marshall/v4/ProducerAckMarshaller.h . .	3842
src/main/activemq/wireformat/openwire/marshall/v4/ProducerIdMarshaller.h . . .	3847
src/main/activemq/wireformat/openwire/marshall/v4/ProducerInfoMarshaller.h . .	3852
src/main/activemq/wireformat/openwire/marshall/v4/RemoveInfoMarshaller.h . .	3857
src/main/activemq/wireformat/openwire/marshall/v4/RemoveSubscriptionInfoMarshaller.h	
3862	
src/main/activemq/wireformat/openwire/marshall/v4/ReplayCommandMarshaller.h	
3867	
src/main/activemq/wireformat/openwire/marshall/v4/ResponseMarshaller.h	3872
src/main/activemq/wireformat/openwire/marshall/v4/SessionIdMarshaller.h	3877
src/main/activemq/wireformat/openwire/marshall/v4/SessionInfoMarshaller.h . . .	3882
src/main/activemq/wireformat/openwire/marshall/v4/ShutdownInfoMarshaller.h .	3887
src/main/activemq/wireformat/openwire/marshall/v4/SubscriptionInfoMarshaller.h	
3892	
src/main/activemq/wireformat/openwire/marshall/v4/TransactionIdMarshaller.h .	3897
src/main/activemq/wireformat/openwire/marshall/v4/TransactionInfoMarshaller.h	3902
src/main/activemq/wireformat/openwire/marshall/v4/WireFormatInfoMarshaller.h	3907
src/main/activemq/wireformat/openwire/marshall/v4/XATransactionIdMarshaller.h	
3912	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBlobMessageMarshaller.h	
3608	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBytesMessageMarshaller.h	
3613	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQDestinationMarshaller.h	
3618	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMapMessageMarshaller.h	
3623	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMessageMarshaller.h	
3628	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQObjectMessageMarshaller.h	
3633	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQQueueMarshaller.h	
3638	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQStreamMessageMarshaller.h	
3643	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempDestinationMarshaller.h	
3648	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempQueueMarshaller.h	
3653	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempTopicMarshaller.h	
3658	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextMessageMarshaller.h	
3663	
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTopicMarshaller.h	3668
src/main/activemq/wireformat/openwire/marshall/v5/BaseCommandMarshaller.h	3673
src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarshaller.h	3678
src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMarshaller.h . . .	3683

src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionControlMarshaller.h	
	3688	
src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionErrorMarshaller.h	
	3693	
src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionIdMarshaller.h	3698
src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionInfoMarshaller.h	3703
src/main/activemq/wireformat/openwire/marshal/v5/	ConsumerControlMarshaller.h	
	3708	
src/main/activemq/wireformat/openwire/marshal/v5/	ConsumerIdMarshaller.h	3713
src/main/activemq/wireformat/openwire/marshal/v5/	ConsumerInfoMarshaller.h	3718
src/main/activemq/wireformat/openwire/marshal/v5/	ControlCommandMarshaller.h	
	3723	
src/main/activemq/wireformat/openwire/marshal/v5/	DataArrayResponseMarshaller.h	
	3728	
src/main/activemq/wireformat/openwire/marshal/v5/	DataResponseMarshaller.h	3733
src/main/activemq/wireformat/openwire/marshal/v5/	DestinationInfoMarshaller.h	3738
src/main/activemq/wireformat/openwire/marshal/v5/	DiscoveryEventMarshaller.h	3743
src/main/activemq/wireformat/openwire/marshal/v5/	ExceptionResponseMarshaller.h	
	3748	
src/main/activemq/wireformat/openwire/marshal/v5/	FlushCommandMarshaller.h	3753
src/main/activemq/wireformat/openwire/marshal/v5/	IntegerResponseMarshaller.h	
	3758	
src/main/activemq/wireformat/openwire/marshal/v5/	JournalQueueAckMarshaller.h	
	3763	
src/main/activemq/wireformat/openwire/marshal/v5/	JournalTopicAckMarshaller.h	
	3768	
src/main/activemq/wireformat/openwire/marshal/v5/	JournalTraceMarshaller.h	3773
src/main/activemq/wireformat/openwire/marshal/v5/	JournalTransactionMarshaller.h	
	3778	
src/main/activemq/wireformat/openwire/marshal/v5/	KeepAliveInfoMarshaller.h	3783
src/main/activemq/wireformat/openwire/marshal/v5/	LastPartialCommandMarshaller.h	
	3788	
src/main/activemq/wireformat/openwire/marshal/v5/	LocalTransactionIdMarshaller.h	
	3793	
src/main/activemq/wireformat/openwire/marshal/v5/	MarshallerFactory.h	3798
src/main/activemq/wireformat/openwire/marshal/v5/	MessageAckMarshaller.h	3803
src/main/activemq/wireformat/openwire/marshal/v5/	MessageDispatchMarshaller.h	
	3808	
src/main/activemq/wireformat/openwire/marshal/v5/	MessageDispatchNotificationMarshaller.h	
	3813	
src/main/activemq/wireformat/openwire/marshal/v5/	MessageIdMarshaller.h	3818
src/main/activemq/wireformat/openwire/marshal/v5/	MessageMarshaller.h	3823
src/main/activemq/wireformat/openwire/marshal/v5/	MessagePullMarshaller.h	3828
src/main/activemq/wireformat/openwire/marshal/v5/	NetworkBridgeFilterMarshaller.h	
	3833	
src/main/activemq/wireformat/openwire/marshal/v5/	PartialCommandMarshaller.h	
	3838	
src/main/activemq/wireformat/openwire/marshal/v5/	ProducerAckMarshaller.h	3843
src/main/activemq/wireformat/openwire/marshal/v5/	ProducerIdMarshaller.h	3848
src/main/activemq/wireformat/openwire/marshal/v5/	ProducerInfoMarshaller.h	3853
src/main/activemq/wireformat/openwire/marshal/v5/	RemoveInfoMarshaller.h	3858
src/main/activemq/wireformat/openwire/marshal/v5/	RemoveSubscriptionInfoMarshaller.h	
	3863	

src/main/activemq/wireformat/openwire/marshall/v5/ReplayCommandMarshaller.h	3868
src/main/activemq/wireformat/openwire/marshall/v5/ResponseMarshaller.h	3873
src/main/activemq/wireformat/openwire/marshall/v5/SessionIdMarshaller.h	3878
src/main/activemq/wireformat/openwire/marshall/v5/SessionInfoMarshaller.h . . .	3883
src/main/activemq/wireformat/openwire/marshall/v5/ShutdownInfoMarshaller.h .	3888
src/main/activemq/wireformat/openwire/marshall/v5/SubscriptionInfoMarshaller.h	3893
src/main/activemq/wireformat/openwire/marshall/v5/TransactionIdMarshaller.h .	3898
src/main/activemq/wireformat/openwire/marshall/v5/TransactionInfoMarshaller.h	3903
src/main/activemq/wireformat/openwire/marshall/v5/WireFormatInfoMarshaller.h	3908
src/main/activemq/wireformat/openwire/marshall/v5/XATransactionIdMarshaller.h	3913
src/main/activemq/wireformat/openwire/utils/BooleanStream.h	3918
src/main/activemq/wireformat/openwire/utils/HexTable.h	3919
src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h . . .	3920
src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h	3921
src/main/activemq/wireformat/stomp/StompCommandConstants.h	3922
src/main/activemq/wireformat/stomp/StompFrame.h	3923
src/main/activemq/wireformat/stomp/StompHelper.h	3924
src/main/activemq/wireformat/stomp/StompWireFormat.h	3925
src/main/activemq/wireformat/stomp/StompWireFormatFactory.h	3926
src/main/cms/BytesMessage.h	3931
src/main/cms/Closeable.h	3932
src/main/cms/CMSException.h	3934
src/main/cms/CMSProperties.h	3935
src/main/cms/CMSSecurityException.h	3936
src/main/cms/Config.h	3590
src/main/cms/Connection.h	3937
src/main/cms/ConnectionFactory.h	3938
src/main/cms/ConnectionMetaData.h	3939
src/main/cms/DeliveryMode.h	3940
src/main/cms/Destination.h	3941
src/main/cms/ExceptionListener.h	3942
src/main/cms/IllegalStateException.h	3943
src/main/cms/InvalidClientIdException.h	3945
src/main/cms/InvalidDestinationException.h	3946
src/main/cms/InvalidSelectorException.h	3947
src/main/cms/MapMessage.h	3948
src/main/cms/Message.h	3493
src/main/cms/MessageConsumer.h	3949
src/main/cms/MessageEOFException.h	3950
src/main/cms/MessageFormatException.h	3951
src/main/cms/MessageListener.h	3952
src/main/cms/MessageNotReadableException.h	3953
src/main/cms/MessageNotWritableException.h	3954
src/main/cms/MessageProducer.h	3955
src/main/cms/ObjectMessage.h	3956
src/main/cms/Queue.h	3957
src/main/cms/QueueBrowser.h	3959
src/main/cms/Session.h	3960
src/main/cms/Startable.h	3961
src/main/cms/Stopable.h	3962
src/main/cms/StreamMessage.h	3963

src/main/cms/ TemporaryQueue.h	3964
src/main/cms/ TemporaryTopic.h	3965
src/main/cms/ TextMessage.h	3966
src/main/cms/ Topic.h	3967
src/main/cms/ UnsupportedOperationException.h	3968
src/main/decaf/internal/ AprPool.h	3970
src/main/decaf/internal/ DecafRuntime.h	3971
src/main/decaf/internal/io/ StandardErrorOutputStream.h	3972
src/main/decaf/internal/io/ StandardInputStream.h	3973
src/main/decaf/internal/io/ StandardOutputStream.h	3974
src/main/decaf/internal/net/ URIEncoderDecoder.h	3975
src/main/decaf/internal/net/ URIHelper.h	3976
src/main/decaf/internal/net/ URIType.h	3977
src/main/decaf/internal/nio/ BufferFactory.h	3978
src/main/decaf/internal/nio/ ByteBuffer.h	3979
src/main/decaf/internal/nio/ ByteBufferPerspective.h	3980
src/main/decaf/internal/nio/ CharArrayBuffer.h	3981
src/main/decaf/internal/nio/ DoubleArrayBuffer.h	3982
src/main/decaf/internal/nio/ FloatArrayBuffer.h	3983
src/main/decaf/internal/nio/ IntArrayBuffer.h	3984
src/main/decaf/internal/nio/ LongArrayBuffer.h	3985
src/main/decaf/internal/nio/ ShortArrayBuffer.h	3986
src/main/decaf/internal/util/ ByteArrayAdapter.h	3987
src/main/decaf/internal/util/ HexStringParser.h	3998
src/main/decaf/internal/util/ TimerTaskHeap.h	3999
src/main/decaf/internal/util/concurrent/ ConditionImpl.h	3988
src/main/decaf/internal/util/concurrent/ MutexImpl.h	3989
src/main/decaf/internal/util/concurrent/ SynchronizableImpl.h	3990
src/main/decaf/internal/util/concurrent/ Transferer.h	3991
src/main/decaf/internal/util/concurrent/ TransferQueue.h	3992
src/main/decaf/internal/util/concurrent/ TransferStack.h	3993
src/main/decaf/internal/util/concurrent/unix/ ConditionHandle.h	3994
src/main/decaf/internal/util/concurrent/unix/ MutexHandle.h	3996
src/main/decaf/internal/util/concurrent/windows/ ConditionHandle.h	3995
src/main/decaf/internal/util/concurrent/windows/ MutexHandle.h	3997
src/main/decaf/io/ BlockingByteArrayInputStream.h	4000
src/main/decaf/io/ BufferedInputStream.h	4001
src/main/decaf/io/ BufferedOutputStream.h	4002
src/main/decaf/io/ ByteArrayInputStream.h	4003
src/main/decaf/io/ ByteArrayOutputStream.h	4004
src/main/decaf/io/ Closeable.h	3933
src/main/decaf/io/ DataInputStream.h	4005
src/main/decaf/io/ DataOutputStream.h	4006
src/main/decaf/io/ EOFException.h	4007
src/main/decaf/io/ FilterInputStream.h	4008
src/main/decaf/io/ FilterOutputStream.h	4009
src/main/decaf/io/ InputStream.h	4010
src/main/decaf/io/ InterruptedIOException.h	4011
src/main/decaf/io/ IOException.h	4012
src/main/decaf/io/ OutputStream.h	4013
src/main/decaf/io/ Reader.h	4014
src/main/decaf/io/ UnsupportedEncodingException.h	4015
src/main/decaf/io/ UTFDataFormatException.h	4016
src/main/decaf/io/ Writer.h	4017

src/main/decaf/lang/	Appendable.h	4018
src/main/decaf/lang/	Boolean.h	4019
src/main/decaf/lang/	Byte.h	4020
src/main/decaf/lang/	Character.h	4021
src/main/decaf/lang/	CharSequence.h	4022
src/main/decaf/lang/	Comparable.h	4023
src/main/decaf/lang/	Double.h	4024
src/main/decaf/lang/	Exception.h	4025
src/main/decaf/lang/	Float.h	4037
src/main/decaf/lang/	Integer.h	4038
src/main/decaf/lang/	Iterable.h	4039
src/main/decaf/lang/	Long.h	4040
src/main/decaf/lang/	Math.h	4041
src/main/decaf/lang/	Number.h	4042
src/main/decaf/lang/	Pointer.h	4043
src/main/decaf/lang/	Runnable.h	4045
src/main/decaf/lang/	Runtime.h	4046
src/main/decaf/lang/	Short.h	4047
src/main/decaf/lang/	System.h	4048
src/main/decaf/lang/	Thread.h	4049
src/main/decaf/lang/	ThreadGroup.h	4050
src/main/decaf/lang/	Throwable.h	4051
src/main/decaf/lang/exceptions/	ClassCastException.h	4026
src/main/decaf/lang/exceptions/	ExceptionDefines.h	3536
src/main/decaf/lang/exceptions/	IllegalArgumentException.h	4027
src/main/decaf/lang/exceptions/	IllegalMonitorStateException.h	4028
src/main/decaf/lang/exceptions/	IllegalStateException.h	3944
src/main/decaf/lang/exceptions/	IllegalThreadStateException.h	4029
src/main/decaf/lang/exceptions/	IndexOutOfBoundsException.h	4030
src/main/decaf/lang/exceptions/	InterruptedException.h	4031
src/main/decaf/lang/exceptions/	InvalidStateException.h	4032
src/main/decaf/lang/exceptions/	NoSuchElementException.h	4033
src/main/decaf/lang/exceptions/	NullPointerException.h	4034
src/main/decaf/lang/exceptions/	NumberFormatException.h	4035
src/main/decaf/lang/exceptions/	RuntimeException.h	4036
src/main/decaf/lang/exceptions/	UnsupportedOperationException.h	3969
src/main/decaf/net/	BindException.h	4052
src/main/decaf/net/	BufferedSocket.h	4053
src/main/decaf/net/	ConnectException.h	4054
src/main/decaf/net/	HttpRetryException.h	4055
src/main/decaf/net/	MalformedURLException.h	4056
src/main/decaf/net/	NoRouteToHostException.h	4057
src/main/decaf/net/	PortUnreachableException.h	4058
src/main/decaf/net/	ProtocolException.h	4059
src/main/decaf/net/	ServerSocket.h	4060
src/main/decaf/net/	Socket.h	4061
src/main/decaf/net/	SocketError.h	4062
src/main/decaf/net/	SocketException.h	4063
src/main/decaf/net/	SocketFactory.h	4064
src/main/decaf/net/	SocketInputStream.h	4065
src/main/decaf/net/	SocketOutputStream.h	4066
src/main/decaf/net/	SocketTimeoutException.h	4067
src/main/decaf/net/	TcpSocket.h	4068
src/main/decaf/net/	UnknownHostException.h	4069

src/main/decaf/net/UnknownServiceException.h	4070
src/main/decaf/net/URI.h	4071
src/main/decaf/net/URISyntaxException.h	4072
src/main/decaf/net/URL.h	4073
src/main/decaf/net/URLDecoder.h	4074
src/main/decaf/net/URLEncoder.h	4075
src/main/decaf/nio/Buffer.h	4076
src/main/decaf/nio/BufferOverflowException.h	4077
src/main/decaf/nio/BufferUnderflowException.h	4078
src/main/decaf/nio/ByteBuffer.h	4079
src/main/decaf/nio/CharBuffer.h	4080
src/main/decaf/nio/DoubleBuffer.h	4081
src/main/decaf/nio/FloatBuffer.h	4082
src/main/decaf/nio/IntBuffer.h	4083
src/main/decaf/nio/InvalidMarkException.h	4084
src/main/decaf/nio/LongBuffer.h	4085
src/main/decaf/nio/ReadOnlyBufferException.h	4086
src/main/decaf/nio/ShortBuffer.h	4087
src/main/decaf/security/GeneralSecurityException.h	4096
src/main/decaf/security/InvalidKeyException.h	4097
src/main/decaf/security/Key.h	4098
src/main/decaf/security/KeyException.h	4099
src/main/decaf/security/NoSuchAlgorithmException.h	4100
src/main/decaf/security/NoSuchProviderException.h	4101
src/main/decaf/security/Principal.h	4102
src/main/decaf/security/PublicKey.h	4103
src/main/decaf/security/SignatureException.h	4104
src/main/decaf/security/auth/x500/X500Principal.h	4088
src/main/decaf/security/cert/Certificate.h	4089
src/main/decaf/security/cert/CertificateEncodingException.h	4090
src/main/decaf/security/cert/CertificateException.h	4091
src/main/decaf/security/cert/CertificateExpiredException.h	4092
src/main/decaf/security/cert/CertificateNotYetValidException.h	4093
src/main/decaf/security/cert/CertificateParsingException.h	4094
src/main/decaf/security/cert/X509Certificate.h	4095
src/main/decaf/security_provider/SecurityProvider.h	4105
src/main/decaf/security_provider/SecurityProviderMap.h	4106
src/main/decaf/security_provider/SecurityProviderRegistrar.h	4107
src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h	4108
src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h	4109
src/main/decaf/util/AbstractCollection.h	4110
src/main/decaf/util/AbstractList.h	4111
src/main/decaf/util/AbstractMap.h	4112
src/main/decaf/util/AbstractQueue.h	4113
src/main/decaf/util/AbstractSequentialList.h	4114
src/main/decaf/util/AbstractSet.h	4115
src/main/decaf/util/Collection.h	4116
src/main/decaf/util/Comparator.h	4117
src/main/decaf/util/Config.h	3591
src/main/decaf/util/Date.h	4154
src/main/decaf/util/Iterator.h	4155
src/main/decaf/util/List.h	4156
src/main/decaf/util/ListIterator.h	4157
src/main/decaf/util/Map.h	4175

src/main/decaf/util/ PriorityQueue.h	4176
src/main/decaf/util/ Properties.h	4177
src/main/decaf/util/ Queue.h	3958
src/main/decaf/util/ Random.h	4178
src/main/decaf/util/ Set.h	4179
src/main/decaf/util/ StlList.h	4180
src/main/decaf/util/ StlMap.h	4181
src/main/decaf/util/ StlQueue.h	4182
src/main/decaf/util/ StlSet.h	4183
src/main/decaf/util/ StringTokenizer.h	4184
src/main/decaf/util/ Timer.h	4185
src/main/decaf/util/ TimerTask.h	4186
src/main/decaf/util/ UUID.h	4187
src/main/decaf/util/comparators/ Less.h	4118
src/main/decaf/util/concurrent/ BlockingQueue.h	4122
src/main/decaf/util/concurrent/ BrokenBarrierException.h	4123
src/main/decaf/util/concurrent/ Callable.h	4124
src/main/decaf/util/concurrent/ CancellationException.h	4125
src/main/decaf/util/concurrent/ Concurrent.h	4126
src/main/decaf/util/concurrent/ ConcurrentMap.h	4127
src/main/decaf/util/concurrent/ ConcurrentStlMap.h	4128
src/main/decaf/util/concurrent/ CountDownLatch.h	4129
src/main/decaf/util/concurrent/ Delayed.h	4130
src/main/decaf/util/concurrent/ ExecutionException.h	4131
src/main/decaf/util/concurrent/ Executor.h	4132
src/main/decaf/util/concurrent/ ExecutorService.h	4133
src/main/decaf/util/concurrent/ Future.h	4134
src/main/decaf/util/concurrent/ Lock.h	4135
src/main/decaf/util/concurrent/ Mutex.h	4141
src/main/decaf/util/concurrent/ PooledThread.h	4142
src/main/decaf/util/concurrent/ PooledThreadListener.h	4143
src/main/decaf/util/concurrent/ RejectedExecutionException.h	4144
src/main/decaf/util/concurrent/ RejectedExecutionHandler.h	4145
src/main/decaf/util/concurrent/ Semaphore.h	4146
src/main/decaf/util/concurrent/ Synchronizable.h	4147
src/main/decaf/util/concurrent/ SynchronousQueue.h	4148
src/main/decaf/util/concurrent/ TaskListener.h	4149
src/main/decaf/util/concurrent/ ThreadFactory.h	4150
src/main/decaf/util/concurrent/ ThreadPool.h	4151
src/main/decaf/util/concurrent/ TimeoutException.h	4152
src/main/decaf/util/concurrent/ TimeUnit.h	4153
src/main/decaf/util/concurrent/atomic/ AtomicBoolean.h	4119
src/main/decaf/util/concurrent/atomic/ AtomicInteger.h	4120
src/main/decaf/util/concurrent/atomic/ AtomicReference.h	4121
src/main/decaf/util/concurrent/locks/ Condition.h	4137
src/main/decaf/util/concurrent/locks/ Lock.h	4136
src/main/decaf/util/concurrent/locks/ LockSupport.h	4138
src/main/decaf/util/concurrent/locks/ ReadWriteLock.h	4139
src/main/decaf/util/concurrent/locks/ ReentrantLock.h	4140
src/main/decaf/util/logging/ ConsoleHandler.h	4158
src/main/decaf/util/logging/ Filter.h	4159
src/main/decaf/util/logging/ Formatter.h	4160
src/main/decaf/util/logging/ Handler.h	4161
src/main/decaf/util/logging/ Logger.h	4162

src/main/decaf/util/logging/ LoggerCommon.h	4163
src/main/decaf/util/logging/ LoggerDefines.h	4164
src/main/decaf/util/logging/ LoggerHierarchy.h	4166
src/main/decaf/util/logging/ LogManager.h	4167
src/main/decaf/util/logging/ LogRecord.h	4168
src/main/decaf/util/logging/ LogWriter.h	4169
src/main/decaf/util/logging/ MarkBlockLogger.h	4170
src/main/decaf/util/logging/ PropertiesChangeListener.h	4171
src/main/decaf/util/logging/ SimpleFormatter.h	4172
src/main/decaf/util/logging/ SimpleLogger.h	4173
src/main/decaf/util/logging/ StreamHandler.h	4174

Chapter 5

Namespace Documentation

5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.2 activemq::cmsutil Namespace Reference

Data Structures

- class **CachedConsumer**
A cached message consumer contained within a pooled session.
- class **CachedProducer**
A cached message producer contained within a pooled session.
- class **CmsAccessor**
Base class for `activemq.cmsutil.CmsTemplate` (p. 1041) and other CMS-accessing gateway helpers, defining common properties such as the CMS `cms.ConnectionFactory` (p. 1184) to operate on.
- class **CmsDestinationAccessor**
Extends the `CmsAccessor` (p. 1024) to add support for resolving destination names.
- class **CmsTemplate**
`CmsTemplate` (p. 1041) simplifies performing synchronous CMS operations.
- class **DestinationResolver**
Resolves a CMS destination name to a `Destination`.
- class **DynamicDestinationResolver**
Resolves a CMS destination name to a `Destination`.
- class **MessageCreator**
Creates the user-defined message to be sent by the `CmsTemplate` (p. 1041).
- class **PooledSession**
A pooled session object that wraps around a delegate session.
- class **ProducerCallback**
Callback for sending a message to a CMS destination.
- class **ResourceLifecycleManager**
Manages the lifecycle of a set of CMS resources.
- class **SessionCallback**
Callback for executing any number of operations on a provided CMS Session.
- class **SessionPool**
A pool of CMS sessions from the same connection and with the same acknowledge mode.

5.3 activemq::commands Namespace Reference

Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

This class represents an Exception sent from the Broker.

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**
- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**
- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**

- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

5.4 activemq::core Namespace Reference

Data Structures

- class **ActiveMQAckHandler**

Interface class that is used to give CMS Messages an interface to Ack themselves with.

- class **ActiveMQConnection**

Concrete connection used for all connectors to the ActiveMQ broker.

- class **ActiveMQConnectionFactory**

- class **ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.*

- class **ActiveMQConnectionSupport**

- class **ActiveMQConstants**

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

- class **ActiveMQConsumer**

- class **ActiveMQProducer**

- class **ActiveMQSession**

- class **ActiveMQSessionExecutor**

Delegate dispatcher for a single session.

- class **ActiveMQTransactionContext**

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

- class **DispatchData**

Simple POJO that contains the information necessary to route a message to a specified consumer.

- class **Dispatcher**

Interface for an object responsible for dispatching messages to consumers.

- class **MessageDispatchChannel**

- class **Synchronization**

*Transacted Object **Synchronization** (p. 3137), used to sync the events of a Transaction with the items in the Transaction.*

5.5 activemq::exceptions Namespace Reference

Data Structures

- class **ActiveMQException**
- class **BrokerException**

5.6 activemq::io Namespace Reference

Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**

OutputStream filter that just logs the data being written.

5.7 activemq::library Namespace Reference

Data Structures

- class `ActiveMQCPP`

5.8 activemq::state Namespace Reference

Data Structures

- class **CommandVisitor**

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

- class **CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1069) that returns NULL for all calls.*

- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

5.9 activemq::threads Namespace Reference

Data Structures

- class **CompositeTask**

Represents a single task that can be part of a set of Tasks that are contained in a `CompositeTaskRunner` (p. 1092).

- class **CompositeTaskRunner**

A `Task` (p. 3148) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

- class **DedicatedTaskRunner**

- class **Task**

Represents a unit of work that requires one or more iterations to complete.

- class **TaskRunner**

5.10 activemq::transport Namespace Reference

Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **inactivity**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

Data Structures

- class **AbstractTransportFactory**
*Abstract implementation of the **TransportFactory** (p. 3279) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3279) instances.*
- class **CompositeTransport**
*A Composite **Transport** (p. 3273) is a **Transport** (p. 3273) implementation that is composed of several **Transports**.*
- class **DefaultTransportListener**
- class **IOTransport**
*Implementation of the **Transport** (p. 3273) interface that performs marshaling of commands to IO streams.*
- class **Transport**
Interface for a transport layer for command objects.
- class **TransportFactory**
*Defines the interface for Factories that create **Transports** or **TransportFilters**.*
- class **TransportFilter**
A filter on the transport layer.
- class **TransportListener**
A listener of asynchronous exceptions from a command transport object.
- class **TransportRegistry**
*Registry of all **Transport** (p. 3273) Factories that are available to the client at runtime.*

5.11 activemq::transport::correlator Namespace Reference

Data Structures

- class **FutureResponse**

A container that holds a response object.

- class **ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

5.12 activemq::transport::failover Namespace Reference

Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**

*Creates an instance of a **FailoverTransport** (p. 1612).*

- class **FailoverTransportListener**

*Utility class used by the **Transport** (p. 3273) to perform the work of responding to events from the active **Transport** (p. 3273).*

- class **URIPool**

5.13 activemq::transport::inactivity Namespace Reference

Data Structures

- class **InactivityMonitor**
- class **ReadChecker**
Runnable class that is used by the {}.
- class **WriteChecker**
Runnable class used by the {}.

5.14 activemq::transport::logging Namespace Reference

Data Structures

- class **LoggingTransport**

A transport filter that logs commands as they are sent/received.

5.15 activemq::transport::mock Namespace Reference

Data Structures

- class **InternalCommandListener**

*Listens for Commands sent from the **MockTransport** (p. 2371).*

- class **MockTransport**

*The **MockTransport** (p. 2371) defines a base level **Transport** (p. 3273) class that is intended to be used in place of an a regular protocol **Transport** (p. 3273) such as TCP.*

- class **MockTransportFactory**

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

- class **ResponseBuilder**

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

5.16 activemq::transport::tcp Namespace Reference

Data Structures

- class **TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1823).*

- class **TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 3160).*

5.17 activemq::util Namespace Reference

Data Structures

- class **ActiveMQProperties**

Implementation of the `CMSProperties` interface that delegates to a `decaf::util::Properties` (p. 2657) object.

- class **CMSExceptionSupport**
- class **CompositeData**

Represents a Composite URI.

- class **LongSequenceGenerator**

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

- class **MemoryUsage**
- class **PrimitiveList**

List of primitives.

- class **PrimitiveMap**

Map of named primitives.

- class **PrimitiveValueConverter**

Class controls the conversion of data contained in a `PrimitiveValueNode` (p. 2555) from one type to another.

- class **PrimitiveValueNode**

Class that wraps around a single value of one of the many types.

- class **URISupport**
- class **Usage**

5.18 activemq::wireformat Namespace Reference

Namespaces

- namespace **openwire**
- namespace **stomp**

Data Structures

- class **MarshalAware**
- class **WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.
- class **WireFormatFactory**
*The **WireFormatFactory** (p. 3367) is the interface that all **WireFormatFactory** (p. 3367) classes must extend.*
- class **WireFormatNegotiator**
*Defines a **WireFormatNegotiator** (p. 3399) which allows a **WireFormat** (p. 3363) to.*
- class **WireFormatRegistry**
*Registry of all **WireFormat** (p. 3363) Factories that are available to the client at runtime.*

5.19 activemq::wireformat::openwire Namespace Reference

Namespaces

- namespace **marshal**
- namespace **utils**

Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**

5.20 activemq::wireformat::openwire::marshal Namespace Reference

Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**
- namespace **v4**
- namespace **v5**

Data Structures

- class **BaseDataStreamMarshaller**
Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.
- class **DataStreamMarshaller**
Base class for all classes that marshal commands for Openwire.
- class **PrimitiveTypesMarshaller**
This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

5.21 `activemq::wireformat::openwire::marshal::v1` Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 181).
- class **ActiveMQBytesMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 217).
- class **ActiveMQDestinationMarshaller**
Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 289).
- class **ActiveMQMapMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 326).
- class **ActiveMQMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 349).
- class **ActiveMQObjectMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 390).
- class **ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 427).
- class **ActiveMQStreamMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQStreamMessageMarshaller` (p. 483).
- class **ActiveMQTempDestinationMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTempDestinationMarshaller` (p. 507).
- class **ActiveMQTempQueueMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 532).
- class **ActiveMQTempTopicMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 557).
- class **ActiveMQTextMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 582).
- class **ActiveMQTopicMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 606).
- class **BaseCommandMarshaller**
Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 664).
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 759).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 787).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1144).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1168).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1195).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1225).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1263).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1288).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1313).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1342).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1367).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1402).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1501).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1531).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1589).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1683).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1784).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1850).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1867).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1898).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1922).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1949).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1965).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2009).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2210).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2243).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2269).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2300).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2325).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2359).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2409).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2489).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2599).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2623).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2648).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2707).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2740).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2772).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2806).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2861).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2881).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2937).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3102).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3220).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3245).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3395).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3419).*

5.22 activemq::wireformat::openwire::marshal::v2 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 193).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 338).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 361).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 402).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 439).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 495).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 519).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 544).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 569).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 594).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 618).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 685).*
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 771).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 799).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1156).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1180).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1207).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1233).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1271).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1296).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1325).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1350).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1375).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1414).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1489).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1515).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1585).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1679).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1780).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1838).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1863).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1886).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1910).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1933).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1961).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1997).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2194).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2231).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2257).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2284).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2305).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2351).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2397).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2477).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2583).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2611).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2636).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2711).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2736).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2756).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2791).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2865).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2889).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2949).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3118).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3224).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3261).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3387).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3427).*

5.23 `activemq::wireformat::openwire::marshal::v3` Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 177).
- class **ActiveMQBytesMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 213).
- class **ActiveMQDestinationMarshaller**
Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 285).
- class **ActiveMQMapMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 322).
- class **ActiveMQMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 345).
- class **ActiveMQObjectMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 386).
- class **ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 423).
- class **ActiveMQStreamMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQStreamMessageMarshaller` (p. 479).
- class **ActiveMQTempDestinationMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTempDestinationMarshaller` (p. 503).
- class **ActiveMQTempQueueMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 528).
- class **ActiveMQTempTopicMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 553).
- class **ActiveMQTextMessageMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 578).
- class **ActiveMQTopicMarshaller**
Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 602).
- class **BaseCommandMarshaller**
Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 657).
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 755).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 783).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1140).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1164).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1191).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1217).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1255).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1280).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1309).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1334).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1359).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1398).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1493).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1519).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1593).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1687).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1788).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1846).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1871).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1894).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1914).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1941).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1969).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2001).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2202).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2239).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2265).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2292).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2315).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2355).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2401).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2481).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2587).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2615).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2652).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2715).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2744).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2760).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2796).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2869).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2897).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2945).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3110).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3236).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3253).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3379).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3415).*

5.24 activemq::wireformat::openwire::marshal::v4 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 185).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 330).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 353).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 394).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 431).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 487).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 511).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 536).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 561).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 586).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 610).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 671).*
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 763).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 791).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1148).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1172).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1199).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1221).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1259).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1284).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1317).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1338).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1363).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1410).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1497).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1523).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1597).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1691).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1792).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1842).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1875).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1890).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1918).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1945).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1973).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2005).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2206).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2235).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2261).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2288).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2310).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2367).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2413).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2485).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2591).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2627).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2644).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2719).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2748).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2768).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2811).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2873).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2885).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2941).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3106).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3228).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3249).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3391).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3423).*

5.25 activemq::wireformat::openwire::marshal::v5 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 334).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 357).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 398).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 435).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 491).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 515).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 540).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 565).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 590).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 614).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 678).*
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 767).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 795).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1152).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1176).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1203).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1229).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1267).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1292).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1321).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1346).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1371).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1406).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1505).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1527).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1601).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1695).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1796).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1854).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1879).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1902).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1926).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1937).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1977).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2013).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2198).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2247).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2273).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2296).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2320).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2363).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2405).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2493).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2595).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2619).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2640).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2723).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2732).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2764).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2801).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2857).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2893).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2953).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3114).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3232).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3257).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3383).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3431).*

5.26 activemq::wireformat::openwire::utils Namespace Reference

Data Structures

- class **BooleanStream**

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

- class **HexTable**

*The **HexTable** (p. 1715) class maps hexadecimal strings to the value of an index into the table, i.e.*

- class **MessagePropertyInterceptor**

Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWireMessage` properties.

- class **OpenwireStringSupport**

5.27 activemq::wireformat::stomp Namespace Reference

Data Structures

- class **StompCommandConstants**
- class **StompFrame**

A Stomp-level message frame that encloses all messages to and from the broker.

- class **StompHelper**

Utility Methods used when marshaling to and from StompFrame's.

- class **StompWireFormat**
- class **StompWireFormatFactory**

Factory used to create the Stomp Wire Format instance.

5.28 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Data Structures

- class **BytesMessage**

*A **BytesMessage** (p. 938) object is used to send a message containing a stream of unsigned bytes.*

- class **Closeable**

Interface for a class that implements the close method.

- class **CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.

- class **CMSProperties**

Interface for a Java-like properties object.

- class **CMSSecurityException**

This exception must be thrown when a provider rejects a user name/password submitted by a client.

- class **Connection**

The client's connection to its provider.

- class **ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1131) objects returned implement the CMS **Connection** (p. 1131) interface and hide the CMS Provider specific implementation details behind that interface.*

- class **ConnectionMetaData**

*A **ConnectionMetaData** (p. 1237) object provides information describing the **Connection** (p. 1131) object.*

- class **DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

- class **Destination**

*A **Destination** (p. 1480) object encapsulates a provider-specific address.*

- class **ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1581) that is registered with the **Connection** (p. 1131).*

- class **IllegalStateException**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

- class **InvalidClientIdException**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

- class **InvalidDestinationException**

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

- class **InvalidSelectorException**

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

- class **MapMessage**

*A **MapMessage** (p. 2106) object is used to send a set of name-value pairs.*

- class **Message**

Root of all messages.

- class **MessageConsumer**

*A client uses a **MessageConsumer** (p. 2214) to received messages from a destination.*

- class **MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3081) or **BytesMessage** (p. 938) is being read.*

- class **MessageFormatException**

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

- class **MessageListener**

*A **MessageListener** (p. 2304) object is used to receive asynchronously delivered messages.*

- class **MessageNotReadableException**

This exception must be thrown when a CMS client attempts to read a write-only message.

- class **MessageNotWriteableException**

This exception must be thrown when a CMS client attempts to write to a read-only message.

- class **MessageProducer**

*A client uses a **MessageProducer** (p. 2332) object to send messages to a **Destination** (p. 1480).*

- class **ObjectMessage**

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

- class **Queue**

An interface encapsulating a provider-specific queue name.

- class **QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 2674) without removing them.*

- class **Session**
*A **Session** (p. 2839) object is a single-threaded context for producing and consuming messages.*
- class **Startable**
Interface for a class that implements the start method.
- class **Stoppable**
Interface for a class that implements the stop method.
- class **StreamMessage**
*Interface for a **StreamMessage** (p. 3081).*
- class **TemporaryQueue**
*Defines a Temporary **Queue** (p. 2674) based **Destination** (p. 1480).*
- class **TemporaryTopic**
*Defines a Temporary **Topic** (p. 3215) based **Destination** (p. 1480).*
- class **TextMessage**
Interface for a text message.
- class **Topic**
An interface encapsulating a provider-specific topic name.
- class **UnsupportedOperationException**
This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

5.28.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.29 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **security_provider**
- namespace **util**

5.29.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.30 decaf::internal Namespace Reference

Namespaces

- namespace **io**
- namespace **net**
- namespace **nio**
- namespace **util**

Data Structures

- class **AprPool**
Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.
- class **DecafRuntime**
Handles APR initialization and termination.

5.31 decaf::internal::io Namespace Reference

Data Structures

- class **StandardErrorOutputStream**

Wrapper Around the Standard error Output facility on the current platform.

- class **StandardInputStream**
- class **StandardOutputStream**

5.32 decaf::internal::net Namespace Reference

Data Structures

- class **URIEncoderDecoder**
- class **URIHelper**

Helper class used by the URI classes in encoding and decoding of URI's.

- class **URIType**

Basic type object that holds data that composes a given URI.

5.33 decaf::internal::nio Namespace Reference

Data Structures

- class **BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 128) package to create the various default version of the NIO interfaces.*

- class **ByteBuffer**

This class defines six categories of operations upon byte buffers:.

- class **ByteArrayPerspective**

*This class extends **ByteArray** to create a reference counted byte array that can be held and used by several different **ByteBuffers** and allow them to know on destruction whose job it is to delete the perspective.*

- class **CharArrayBuffer**
- class **DoubleArrayBuffer**
- class **FloatArrayBuffer**
- class **IntArrayBuffer**
- class **LongArrayBuffer**
- class **ShortArrayBuffer**

5.34 decaf::internal::util Namespace Reference

Namespaces

- namespace **concurrent**

Data Structures

- class **ByteArrayAdapter**

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

- class **HexStringParser**
- class **TimerTaskHeap**

A Binary Heap implemented specifically for the Timer class in Decaf Util.

5.35 decaf::internal::util::concurrent Namespace Reference

Data Structures

- class **ConditionImpl**
- class **MutexImpl**
- class **SynchronizableImpl**

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

- class **Transferer**

Shared internal API for dual stacks and queues.

- class **TransferQueue**

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

- class **TransferStack**

5.36 decaf::io Namespace Reference

Data Structures

- class **BlockingByteArrayInputStream**
This is a blocking version of a byte buffer stream.
- class **BufferedInputStream**
A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.
- class **BufferedOutputStream**
Wrapper around another output stream that buffers output before writing to the target output stream.
- class **ByteArrayInputStream**
*Simple implementation of **InputStream** (p. 1740) that wraps around an STL Vector `std::vector<unsigned char>`.*
- class **ByteArrayOutputStream**
- class **Closeable**
Interface for a class that implements the close method.
- class **DataInputStream**
A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.
- class **DataOutputStream**
A data output stream lets an application write primitive Java data types to an output stream in a portable way.
- class **EOFException**
- class **FilterInputStream**
*A **FilterInputStream** (p. 1631) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*
- class **FilterOutputStream**
This class is the superclass of all classes that filter output streams.
- class **InputStream**
Base interface for an input stream.
- class **InterruptedIOException**
- class **IOException**
- class **OutputStream**
Base interface for an output stream.
- class **Reader**
- class **UnsupportedEncodingException**
Thrown when the the Character Encoding is not supported.

- class **UTFDataFormatException**

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

- class **Writer**

5.37 decaf::lang Namespace Reference

Namespaces

- namespace **exceptions**

Data Structures

- class **Appendable**
- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**

*A **CharSequence** (p. 1014) is a readable sequence of char values.*

- class **Comparable**

This interface imposes a total ordering on the objects of each class that implements it.

- class **Double**
- class **Exception**
- class **Float**
- class **Integer**
- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 1830) type for generic collections API calls.*

- class **Long**
- class **Math**

*The class **Math** (p. 2126) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 2432) is the superclass of classes **Byte** (p. 840), **Double** (p. 1537), **Float** (p. 1647), **Integer** (p. 1762), **Long** (p. 2057), and **Short** (p. 2906).*

- class **AtomicRefCounter**
- struct **STATIC_CAST_TOKEN**
- struct **DYNAMIC_CAST_TOKEN**
- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 2497) that is a template on a Type and is Thread Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2497) instance.*

- class **Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

- class **Runtime**
- class **Short**
- class **System**
- class **ThreadGroup**
- class **Throwable**

This class represents an error that has occurred.

Functions

- template<typename T , typename R , typename U >
bool **operator**== (const **Pointer**< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **operator**== (const U *left, const **Pointer**< T, R > &right)
- template<typename T , typename R , typename U >
bool **operator**!= (const **Pointer**< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **operator**!= (const U *left, const **Pointer**< T, R > &right)

5.37.1 Function Documentation

5.37.1.1 template<typename T , typename R , typename U > bool
 decaf::lang::operator!= (const U * *left*, const **Pointer**< T, R > & *right*)
 [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.37.1.2 template<typename T , typename R , typename U > bool
 decaf::lang::operator!= (const **Pointer**< T, R > & *left*, const U * *right*)
 [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.37.1.3 template<typename T , typename R , typename U > bool
 decaf::lang::operator== (const U * *left*, const **Pointer**< T, R > & *right*)
 [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.37.1.4 template<typename T , typename R , typename U > bool
 decaf::lang::operator== (const **Pointer**< T, R > & *left*, const U * *right*)
 [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.38 decaf::lang::exceptions Namespace Reference

Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IllegalThreadStateException**
- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NoSuchElementException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**

5.39 decaf::net Namespace Reference

Data Structures

- class **BindException**
- class **BufferedSocket**

*Buffered **Socket** (p. 2964) class that wraps a **Socket** (p. 2964) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.*

- class **ConnectException**
- class **HttpRetryException**
- class **MalformedURLErrorException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

A server socket class (for testing purposes).

- class **Socket**
- class **SocketError**

Static utility class to simplify handling of error codes for socket operations.

- class **SocketException**

Exception for errors when manipulating sockets.

- class **SocketFactory**

***Socket** (p. 2964) Factory implementation for use in Creating Sockets.*

- class **SocketInputStream**

Input stream for performing reads on a socket.

- class **SocketOutputStream**

Output stream for performing write operations on a socket.

- class **SocketTimeoutException**
- class **TcpSocket**

Platform-independent implementation of the socket interface.

- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**

*This class represents an instance of a **URI** (p. 3308) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

*Class **URL** (p. 3347) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

5.40 decaf::nio Namespace Reference

Data Structures

- class **Buffer**

A container for data of a specific primitive type.

- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**

This class defines six categories of operations upon byte buffers:.

- class **CharBuffer**

This class defines four categories of operations upon character buffers:.

- class **DoubleBuffer**

This class defines four categories of operations upon double buffers:.

- class **FloatBuffer**

This class defines four categories of operations upon float buffers:.

- class **IntBuffer**

This class defines four categories of operations upon int buffers:.

- class **InvalidMarkException**

- class **LongBuffer**

This class defines four categories of operations upon long long buffers:.

- class **ReadOnlyBufferException**

- class **ShortBuffer**

This class defines four categories of operations upon short buffers:.

5.41 decaf::security Namespace Reference

Namespaces

- namespace **auth**
- namespace **cert**

Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 1953) interface is the top-level interface for all keys.*

- class **KeyException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

Base interface for a principal, which can represent an individual or organization.

- class **PublicKey**

A public key.

- class **SignatureException**

5.42 decaf::security::auth Namespace Reference

Namespaces

- namespace **x500**

5.43 decaf::security::auth::x500 Namespace Reference

Data Structures

- class **X500Principal**

5.44 decaf::security::cert Namespace Reference

Data Structures

- class **Certificate**
Base interface for all identity certificates.
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**
Base interface for all identity certificates.

5.45 decaf::security_provider Namespace Reference

Namespaces

- namespace **unix**

Data Structures

- class **SecurityProvider**
- class **SecurityProviderMap**

Lookup Map for Connector Factories.

- class **SecurityProviderRegistrar**

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

5.46 decaf::security_provider::unix Namespace Reference

Namespaces

- namespace `openssl`

5.47 decaf::security_provider::unix::openssl Namespace Reference

Data Structures

- class **OpenSSLX500Principal**
The `OpenSSLX500Principal` (p. 2439) wraps around an `OpenSSL X509_NAME` structure.
- class **OpenSSLX509Certificate**

5.48 decaf::util Namespace Reference

Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**

Data Structures

- class **AbstractCollection**

*This class provides a skeletal implementation of the **Collection** (p. 1054) interface, to minimize the effort required to implement this interface.*

- class **AbstractList**

*This class provides a skeletal implementation of the **List** (p. 1984) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

- class **AbstractMap**

*This class provides a skeletal implementation of the **Map** (p. 2094) interface, to minimize the effort required to implement this interface.*

- class **AbstractQueue**

*This class provides skeletal implementations of some **Queue** (p. 2671) operations.*

- class **AbstractSequentialList**

*This class provides a skeletal implementation of the **List** (p. 1984) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

- class **AbstractSet**

*This class provides a skeletal implementation of the **Set** (p. 2905) interface to minimize the effort required to implement this interface.*

- class **Collection**

The root interface in the collection hierarchy.

- class **Comparator**

A comparison function, which imposes a total ordering on some collection of objects.

- class **Date**

Wrapper class around a time value in milliseconds.

- class **Iterator**

Defines an object that can be used to iterate over the elements of a collection.

- class **List**

An ordered collection (also known as a sequence).

- class **ListIterator**

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

- class **Map**

***Map** (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **PriorityQueue**

An unbounded priority queue based on a binary heap algorithm.

- class **Properties**

Java-like properties class for mapping string names to string values.

- class **Queue**

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

- class **Random**

***Random** (p. 2677) Value Generator which is used to generate a stream of pseudorandom numbers.*

- class **Set**

A collection that contains no duplicate elements.

- class **StlList**

***List** (p. 1984) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*

- class **StlMap**

***Map** (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **StlQueue**

*The **Queue** (p. 2671) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*

- class **StlSet**

***Set** (p. 2905) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*

- class **StringTokenizer**

- class **Timer**

A facility for threads to schedule tasks for future execution in a background thread.

- class **TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3188).*

- class **UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3356)).*

5.49 decaf::util::comparators Namespace Reference

Data Structures

- class **Less**

*Simple **Less** (p. 1981) **Comparator** (p. 1086) that compares to elements to determine if the first is less than the second.*

5.50 decaf::util::concurrent Namespace Reference

Namespaces

- namespace **atomic**
- namespace **locks**

Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **BrokenBarrierException**
- class **Callable**

A task that returns a result and may throw an exception.

- class **CancellationException**
- class **ConcurrentMap**

*Interface for a **Map** (p. 2094) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2094) interface.*

- class **ConcurrentStlMap**

***Map** (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

- class **CountDownLatch**
- class **Delayed**

A mix-in style interface for marking objects that should be acted upon after a given delay.

- class **ExecutionException**
- class **Executor**

*An object that executes submitted **decaf.lang Runnable** (p. 2816) tasks.*

- class **ExecutorService**

*An **Executor** (p. 1608) that provides methods to manage termination and methods that can produce a **Future** (p. 1701) for tracking progress of one or more asynchronous tasks.*

- class **Future**

*A **Future** (p. 1701) represents the result of an asynchronous computation.*

- class **Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

- class **Mutex**

***Mutex** (p. 2385) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

- class **PooledThread**
- class **PooledThreadListener**

*Abstract Listener Interface for users of **ThreadPool** (p. 3175).*

- class **RejectedExecutionException**
- class **RejectedExecutionHandler**

*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).*

- class **Semaphore**

A counting semaphore.

- class **Synchronizable**

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

- class **SynchronousQueue**

*A **BlockingQueue** blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*

- class **TaskListener**

- class **ThreadFactory**

*public interface **ThreadFactory** (p. 3172)*

- class **ThreadPool**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

- class **TimeoutException**

- class **TimeUnit**

*A **TimeUnit** (p. 3205) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

5.51 decaf::util::concurrent::atomic Namespace Reference

Data Structures

- class **AtomicBoolean**
A boolean value that may be updated atomically.
- class **AtomicInteger**
An int value that may be updated atomically.
- class **AtomicReference**
An Pointer reference that may be updated atomically.

5.52 decaf::util::concurrent::locks Namespace Reference

Data Structures

- class **Condition**

***Condition** (p. 1118) factors out the **Mutex** (p. 2385) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2017) implementations.*

- class **Lock**

***Lock** (p. 2017) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

- class **LockSupport**

Basic thread blocking primitives for creating locks and other synchronization classes.

- class **ReadWriteLock**

*A **ReadWriteLock** (p. 2688) maintains a pair of associated locks, one for read-only operations and one for writing.*

- class **ReentrantLock**

*A reentrant mutual exclusion **Lock** (p. 2017) with extended capabilities.*

5.53 decaf::util::logging Namespace Reference

Data Structures

- class **Filter**

A **Filter** (p. 1630) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

- class **Formatter**

A **Formatter** (p. 1699) provides support for formatting *LogRecords*.

- class **Handler**

A **Handler** (p. 1709) object takes log messages from a **Logger** (p. 2028) and exports them.

- class **Logger**

- class **LoggerHierarchy**

- class **LogManager**

There is a single global **LogManager** (p. 2045) object that is used to maintain a set of shared state about *Loggers* and log services.

- class **LogRecord**

- class **LogWriter**

- class **MarkBlockLogger**

Defines a class that can be used to mark the entry and exit from scoped blocks.

- class **PropertiesChangeListener**

Defines the interface that classes can use to listen for change events on **Properties** (p. 2657).

- class **SimpleFormatter**

Print a brief summary of the **LogRecord** (p. 2050) in a human readable format.

- class **SimpleLogger**

- class **StreamHandler**

Enumerations

- enum **Level** {

Off, Null, Markblock, Debug,

Info, Warn, Error, Fatal,

Throwing }

Defines an enumeration for logging levels.

5.53.1 Enumeration Type Documentation

5.53.1.1 enum decaf::util::logging::Level

Defines an enumeration for logging levels.

Enumerator:

Off

Null

Markblock

Debug

Info

Warn

Error

Fatal

Throwing

5.54 std Namespace Reference

Data Structures

- struct `less< decaf::lang::Pointer< T > >`

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Chapter 6

Data Structure Documentation

6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p.1054) interface, to minimize the effort required to implement this interface.

#include <src/main/decaf/util/AbstractCollection.h> Inheritance diagram for decaf::util::AbstractCollection< E >:

Public Member Functions

- virtual ~**AbstractCollection** ()
- **AbstractCollection**< E > & **operator=** (const **AbstractCollection**< E > &collection)
Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.
- virtual bool **add** (const E &value DECAF_UNUSED)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Ensures that this collection contains the specified element (optional operation).
- virtual bool **addAll** (const **Collection**< E > &collection)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Adds all of the elements in the specified collection to this collection (optional operation).
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all of the elements from this collection (optional operation).
- virtual void **copy** (const **Collection**< E > &collection)
*Renders this **Collection** (p.1054) as a Copy of the given **Collection** (p.1054).*

- virtual bool **contains** (const E &value) const throw (lang::Exception)
Returns true if this collection contains the specified element.
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw (lang::Exception)
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (const **Collection**< E > &collection) const
*Answers true if this **Collection** (p. 1054) and the one given are the same size and if each element contained in the **Collection** (p. 1054) given is equal to an element contained in this collection.*
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from this collection, if it is present (optional operation).
- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes all of this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual std::vector< E > **toArray** () const
*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1054).*
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Attributes

- util::concurrent::Mutex mutex

6.1.1 Detailed Description

template<typename E> class decaf::util::AbstractCollection< E >

This class provides a skeletal implementation of the **Collection** (p.1054) interface, to minimize the effort required to implement this interface. To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p.1054) constructor, as per the recommendation in the **Collection** (p.1054) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since:

1.0

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `template<typename E> virtual decaf::util::AbstractCollection< E
>::~~AbstractCollection () [inline, virtual]`

6.1.3 Member Function Documentation

6.1.3.1 `template<typename E> virtual bool decaf::util::AbstractCollection<
E >::add (const E &value DECAF_UNUSED) throw
(lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException,
lang::exceptions::IllegalStateException) [inline,
virtual]`

Ensures that this collection contains the specified element (optional operation). Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p. 1054) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an `UnsupportedOperationException`.

Parameters:

value - The element that must be ensured to be in this collection.

Returns:

true if the collection was changed as a result of this call.

Exceptions:

UnsupportedOperationException if the add operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Referenced by `decaf::util::AbstractCollection< Pointer< BackupTransport > >::addAll()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=()`.


```

6.1.3.2  template<typename E> virtual bool decaf::util::AbstractCollection<
        E >::addAll (const Collection< E > & collection) throw
        ( lang::exceptions::UnsupportedOperationException,
        lang::exceptions::IllegalArgumentException,
        lang::exceptions::IllegalStateException ) [inline,
        virtual]

```

Adds all of the elements in the specified collection to this collection (optional operation). The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

Parameters:

collection - The **Collection** (p. 1054) whose elements are to be added to this **Collection** (p. 1054).

Returns:

true if the collection was changed as a result of this call.

Exceptions:

UnsupportedOperationException if the addAll operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1057).

Reimplemented in **decaf::util::AbstractQueue< E >** (p. 163).

```

6.1.3.3  template<typename E> virtual void decaf::util::AbstractCollection< E
        >::clear () throw ( lang::exceptions::UnsupportedOperationException )
        [inline, virtual]

```

Removes all of the elements from this collection (optional operation). The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1833) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

Implements **decaf::util::Collection< E >** (p. 1057).

Reimplemented in **decaf::util::AbstractQueue< E >** (p. 164), **decaf::util::PriorityQueue< E >** (p. 2574), **decaf::util::StlList< E >** (p. 3024), **decaf::util::StlSet< E >** (p. 3054), **decaf::util::StlList< CompositeTask * >** (p. 3024), **decaf::util::StlList< URI >** (p. 3024), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3024), **decaf::util::StlList< PrimitiveValueNode >** (p. 3024), **decaf::util::StlList< Pointer< Command > >** (p. 3024), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3024), **decaf::util::StlSet< transport::TransportListener * >** (p. 3054), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3054), and **decaf::util::StlSet< ActiveMQSession * >** (p. 3054).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy()**, and **decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=()**.

6.1.3.4 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & value) const throw (lang::Exception)` [inline, virtual]

Returns true if this collection contains the specified element. This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p. 1054).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

Implements **decaf::util::Collection< E >** (p. 1058).

Reimplemented in **decaf::util::StlList< E >** (p. 3024), **decaf::util::StlSet< E >** (p. 3055), **decaf::util::StlList< CompositeTask * >** (p. 3024), **decaf::util::StlList< URI >** (p. 3024), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3024), **decaf::util::StlList< PrimitiveValueNode >** (p. 3024), **decaf::util::StlList< Pointer< Command > >** (p. 3024), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3024), **decaf::util::StlSet< transport::TransportListener * >** (p. 3055), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3055), and **decaf::util::StlSet< ActiveMQSession * >** (p. 3055).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::containsAll()**.

6.1.3.5 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception)` [inline, virtual]

Returns true if this collection contains all of the elements in the specified collection. This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Parameters:

collection collection to be checked for containment in this collection

Returns:

true if this collection contains all of the elements in the specified collection.

Exceptions:

Exception if an error occurs,

Implements **decaf::util::Collection< E >** (p. 1058).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::equals()**.

6.1.3.6 **template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (const Collection< E > & collection) [inline, virtual]**

Renders this **Collection** (p. 1054) as a Copy of the given **Collection** (p. 1054). This implementation iterates over the contents of the given collection adding each to this collection after first calling this Collection's clear method.

Parameters:

collection - the collection to mirror.

6.1.3.7 **template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals (const Collection< E > & collection) const [inline, virtual]**

Answers true if this **Collection** (p. 1054) and the one given are the same size and if each element contained in the **Collection** (p. 1054) given is equal to an element contained in this collection.

Parameters:

collection - The **Collection** (p. 1054) to be compared to this one.

Returns:

true if this **Collection** (p. 1054) is equal to the one given.

Implements **decaf::util::Collection< E >** (p. 1059).

6.1.3.8 **template<typename E> virtual bool decaf::util::AbstractCollection< E >::isEmpty () const [inline, virtual]**

Returns true if this collection contains no elements. This implementation returns **size()** (p. 1062) == 0.

Returns:

true if the size method return 0.

Implements **decaf::util::Collection< E >** (p. 1059).

Reimplemented in **decaf::util::StlList< E >** (p. 3025), **decaf::util::StlSet< E >** (p. 3055), **decaf::util::StlList< CompositeTask * >** (p. 3025), **decaf::util::StlList< URI >** (p. 3025), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3025), **decaf::util::StlList< PrimitiveValueNode >** (p. 3025), **decaf::util::StlList< Pointer< Command > >** (p. 3025),

`decaf::util::StlList< Pointer< BackupTransport > >` (p. 3025), `decaf::util::StlSet< transport::TransportListener * >` (p. 3055), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3055), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3055).

Referenced by `decaf::util::AbstractQueue< E >::clear()`.

6.1.3.9 `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.1.3.10 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

6.1.3.11 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3125).

6.1.3.12 `template<typename E> AbstractCollection<E>& decaf::util::AbstractCollection< E >::operator= (const AbstractCollection< E > & collection) [inline]`

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

Parameters:

collection - the collection to copy

Returns:

a reference to this collection

```
6.1.3.13  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::remove (const E & value) throw (
           lang::exceptions::UnsupportedOperationException,
           lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes a single instance of the specified element from this collection, if it is present (optional operation). More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p.1054).

Implements **decaf::util::Collection< E >** (p.1060).

Reimplemented in **decaf::util::PriorityQueue< E >** (p.2577), **decaf::util::StlList< E >** (p.3027), **decaf::util::StlSet< E >** (p.3056), **decaf::util::StlList< CompositeTask * >** (p.3027), **decaf::util::StlList< URI >** (p.3027), **decaf::util::StlList< Pointer< DestinationInfo > >** (p.3027), **decaf::util::StlList< PrimitiveValueNode >** (p.3027), **decaf::util::StlList< Pointer< Command > >** (p.3027), **decaf::util::StlList< Pointer< BackupTransport > >** (p.3027), **decaf::util::StlSet< transport::TransportListener * >** (p.3056), **decaf::util::StlSet< Pointer< Synchronization > >** (p.3056), and **decaf::util::StlSet< ActiveMQSession * >** (p.3056).

```
6.1.3.14  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::removeAll (const Collection< E > & collection)
           throw ( lang::exceptions::UnsupportedOperationException,
           lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes all of this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the

specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

Parameters:

collection - collection containing elements to be removed from this collection

Returns:

true if this collection changed as a result of the call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Implements `decaf::util::Collection< E >` (p. 1060).

Reimplemented in `decaf::util::AbstractSet< E >` (p. 167), `decaf::util::AbstractSet< transport::TransportListener * >` (p. 167), `decaf::util::AbstractSet< Pointer< Synchronization > >` (p. 167), and `decaf::util::AbstractSet< ActiveMQSession * >` (p. 167).

```
6.1.3.15  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::retainAll (const Collection< E > & collection)
           throw ( lang::exceptions::UnsupportedOperationException,
                   lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

Parameters:

collection - collection containing elements to be retained in this collection

Returns:

true if this collection changed as a result of the call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Implements **decaf::util::Collection< E >** (p. 1061).

6.1.3.16 `template<typename E> virtual std::vector<E>
decaf::util::AbstractCollection< E >::toArray () const [inline, virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1054). All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns:

an vector of copies of all the elements from this **Collection** (p. 1054)

Implements **decaf::util::Collection< E >** (p. 1062).

6.1.3.17 `template<typename E> virtual bool decaf::util::AbstractCollection<
E >::tryLock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.1.3.18 `template<typename E> virtual void decaf::util::AbstractCollection< E
>::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline,
virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.1.3.19 `template<typename E> virtual void decaf::util::AbstractCollection<E >::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3128).

6.1.3.20 `template<typename E> virtual void decaf::util::AbstractCollection<E >::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3130).

6.1.3.21 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3131).

6.1.4 Field Documentation

6.1.4.1 `template<typename E> util::concurrent::Mutex decaf::util::AbstractCollection< E >::mutex [mutable, protected]`

Referenced by `decaf::util::AbstractCollection< Pointer< BackupTransport > >::lock()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notify()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notifyAll()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::tryLock()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::unlock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p.1984) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

#include <src/main/decaf/util/AbstractList.h> Inheritance diagram for decaf::util::AbstractList< E >:

Public Member Functions

- virtual ~**AbstractList** ()

6.2.1 Detailed Description

template<typename E> class decaf::util::AbstractList< E >

This class provides a skeletal implementation of the **List** (p.1984) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), **AbstractSequentialList** (p.166) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the `get(int)` and `size()` (p.1062) methods.

To implement a modifiable list, the programmer must additionally override the `set(int, E)` method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the `add(int, E)` and `remove(int)` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p.1054) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: `get(int)`, `set(int, E)`, `add(int, E)` and `remove(int)`.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since:

1.0

6.2.2 Constructor & Destructor Documentation

6.2.2.1 template<typename E > virtual decaf::util::AbstractList< E >::~~AbstractList () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractList.h**

6.3 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p.2094) interface, to minimize the effort required to implement this interface.

#include <src/main/decaf/util/AbstractMap.h> Inheritance diagram for decaf::util::AbstractMap< K, V, COMPARATOR >:

Public Member Functions

- virtual ~AbstractMap ()

6.3.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::AbstractMap< K, V, COMPARATOR >

This class provides a skeletal implementation of the **Map** (p.2094) interface, to minimize the effort required to implement this interface. To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the entrySet method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p.167). This set should not support the add or remove methods, and its iterator should not support the remove method.

To implement a modifiable map, the programmer must additionally override this class's put method (which otherwise throws an UnsupportedOperationException), and the iterator returned by entrySet().iterator() must additionally implement its remove method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p.2094) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

Since:

1.0

6.3.2 Constructor & Destructor Documentation

6.3.2.1 template<typename K , typename V , typename COMPARATOR >
virtual decaf::util::AbstractMap< K, V, COMPARATOR >::~~AbstractMap
() [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/AbstractMap.h

6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 2671) operations.

#include <src/main/decaf/util/AbstractQueue.h> Inheritance diagram for decaf::util::AbstractQueue< E >:

Public Member Functions

- **AbstractQueue** ()
- virtual ~**AbstractQueue** ()
- virtual bool **add** (const E &value) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.
- virtual bool **addAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Adds all the elements of a collection to the queue.
- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)
Retrieves and removes the head of this queue.
- virtual E **element** () const throw (decaf::lang::exceptions::NoSuchElementException)
Retrieves, but does not remove, the head of this queue.
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all elements of the queue.

6.4.1 Detailed Description

template<typename E> class decaf::util::AbstractQueue< E >

This class provides skeletal implementations of some **Queue** (p. 2671) operations. Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 2671) implementation that extends this class must minimally define a method **Queue** (p. 2671). **offer**(E) which does not permit insertion of null elements, along with methods **Queue** (p. 2671). **peek**() (p. 2672), **Queue.poll**() (p. 2673), **Collection.size**() (p. 1062), and a **Collection.iterator**() (p. 1830) supporting **Iterator.remove**() (p. 1833). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 147).

Since:

1.0

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue()` [inline]

6.4.2.2 `template<typename E > virtual decaf::util::AbstractQueue< E >::~~AbstractQueue()` [inline, virtual]

6.4.3 Member Function Documentation

6.4.3.1 `template<typename E > virtual bool decaf::util::AbstractQueue< E >::add (const E & value) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)` [inline, virtual]

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available. This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

Parameters:

value - the element to offer to the **Queue** (p. 2671).

Returns:

true if the add succeeds.

Exceptions:

IllegalArgumentException if the element cannot be added.

Implements `decaf::util::Collection< E >` (p. 1056).

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 2574).

References `decaf::util::Queue< E >::offer()`.

6.4.3.2 `template<typename E > virtual bool decaf::util::AbstractQueue< E >::addAll (const Collection< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)` [inline, virtual]

Adds all the elements of a collection to the queue. If the collection is the queue itself, then an `IllegalArgumentException` will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

Parameters:

collection - the collection to be added to the queue.

Returns:

true if the operation succeeds.

Exceptions:

IllegalArgumentException If the collection to be added to the queue is the queue itself.

Reimplemented from **decaf::util::AbstractCollection**< E > (p. 151).

6.4.3.3 `template<typename E> virtual void decaf::util::AbstractQueue< E >::clear
() throw (lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all elements of the queue. This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractCollection**< E > (p. 151).

Reimplemented in **decaf::util::PriorityQueue**< E > (p. 2574).

References **decaf::util::AbstractCollection**< E >::isEmpty(), and **decaf::util::Queue**< E >::poll().

6.4.3.4 `template<typename E> virtual E decaf::util::AbstractQueue< E >::element
() const throw (decaf::lang::exceptions::NoSuchElementException)
[inline, virtual]`

Retrieves, but does not remove, the head of this queue. This method differs from peek only in that it throws an exception if this queue is empty.

This implementation returns the result of peek unless the queue is empty.

Returns:

the element in the head of the queue.

Exceptions:

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue**< E > (p. 2672).

References **decaf::util::Queue**< E >::peek().

6.4.3.5 `template<typename E> virtual E decaf::util::AbstractQueue< E >::remove
() throw (decaf::lang::exceptions::NoSuchElementException) [inline,
virtual]`

Retrieves and removes the head of this queue. This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

Returns:

a copy of the element in the head of the queue.

Exceptions:

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2673).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2577).

References **decaf::util::Queue< E >::poll()**.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractQueue.h`

6.5 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p.1984) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

#include <src/main/decaf/util/AbstractSequentialList.h> Inheritance diagram for decaf::util::AbstractSequentialList< E >:

Public Member Functions

- virtual ~AbstractSequentialList ()

6.5.1 Detailed Description

template<typename E> class decaf::util::AbstractSequentialList< E >

This class provides a skeletal implementation of the **List** (p.1984) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), **AbstractList** (p.160) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p.160) class in the sense that it implements the "random access" methods (get(int index), set(int index, E element), add(int index, E element) and remove(int index)) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the listIterator and size methods. For an unmodifiable list, the programmer need only implement the list iterator's hasNext, next, hasPrevious, previous and index methods.

For a modifiable list the programmer should additionally implement the list iterator's set method. For a variable-size list the programmer should additionally implement the list iterator's remove and add methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p.1054) interface specification.

Since:

1.0

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<typename E > virtual decaf::util::AbstractSequentialList< E >::~AbstractSequentialList () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/AbstractSequentialList.h

6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 2905) interface to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractSet.h> Inheritance      diagram      for
decaf::util::AbstractSet< E >:
```

Public Member Functions

- virtual **~AbstractSet** ()
- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes from this set all of its elements that are contained in the specified collection (optional operation).

6.6.1 Detailed Description

```
template<typename E> class decaf::util::AbstractSet< E >
```

This class provides a skeletal implementation of the **Set** (p. 2905) interface to minimize the effort required to implement this interface. The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p. 1054) by extending **AbstractCollection** (p. 147), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 2905) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since:

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 template<typename E> virtual decaf::util::AbstractSet< E >::~AbstractSet () [inline, virtual]

6.6.3 Member Function Documentation

6.6.3.1 template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll (const **Collection**< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes from this set all of its elements that are contained in the specified collection (optional operation). If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates

over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

Parameters:

collection - The **Collection** (p. 1054) whose elements are to be retained

Returns:

true if the collection changed as a result of this call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 155).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSet.h`

6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 3279) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3279) instances.

#include <src/main/activemq/transport/AbstractTransportFactory.h> Inheritance diagram for activemq::transport::AbstractTransportFactory:

Public Member Functions

- virtual **~AbstractTransportFactory** ()

Protected Member Functions

- virtual **Pointer< wireformat::WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw (**decaf::lang::exceptions::NoSuchElementException**)

*Creates the WireFormat that is configured for this **Transport** (p. 3273) and returns it.*

6.7.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 3279) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3279) instances.

Since:

3.0

6.7.2 Constructor & Destructor Documentation

- 6.7.2.1** virtual **activemq::transport::AbstractTransportFactory::~AbstractTransportFactory** () [inline, virtual]

6.7.3 Member Function Documentation

- 6.7.3.1** virtual **Pointer<wireformat::WireFormat> activemq::transport::AbstractTransportFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) throw (**decaf::lang::exceptions::NoSuchElementException**) [protected, virtual]

Creates the WireFormat that is configured for this **Transport** (p.3273) and returns it. The default WireFormat is Openwire.

Parameters:

properties The properties that were configured on the URI.

Returns:

a pointer to a `WireFormat` instance that the caller then owns.

Exceptions:

NoSuchElementException if the configured `WireFormat` is not found.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/AbstractTransportFactory.h`

6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

Public Member Functions

- virtual `~ActiveMQAckHandler ()`
- virtual void `acknowledgeMessage (const commands::Message *message)=0` throw (cms::CMSException)

Method called to acknowledge the message passed.

6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Since:

2.0

6.8.2 Constructor & Destructor Documentation

- 6.8.2.1 virtual `activemq::core::ActiveMQAckHandler::~~ActiveMQAckHandler ()`
[inline, virtual]

6.8.3 Member Function Documentation

- 6.8.3.1 virtual void `activemq::core::ActiveMQAckHandler::acknowledgeMessage (const commands::Message * message)` throw (cms::CMSException)
[pure virtual]

Method called to acknowledge the message passed.

Parameters:

message Message to Acknowledge

Exceptions:

CMSException

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQAckHandler.h`

6.9 activemq::commands::ActiveMQBlobMessage Class Reference

#include <src/main/activemq/commands/ActiveMQBlobMessage.h> Inheritance diagram for activemq::commands::ActiveMQBlobMessage:

Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual **~ActiveMQBlobMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQBlobMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- std::string **getRemoteBlobUrl** () const
Get the Remote URL of the Blob.
- void **setRemoteBlobUrl** (const std::string &remoteURL)
Set the Remote URL of the Blob.
- std::string **getMimeType** () const
Get the Mime Type of the Blob.
- void **setMimeType** (const std::string &mimeType)
Set the Mime Type of the Blob.
- std::string **getName** () const
Gets the Name of the Blob.
- void **setName** (const std::string &name)
Sets the Name of the Blob.

- bool **isDeletedByBroker** () const
Gets if this Blob is deleted by the Broker.
- void **setDeletedByBroker** (bool value)
Sets the Deleted By Broker flag.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY_MIME_TYPE**

6.9.1 Constructor & Destructor Documentation

6.9.1.1 `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ()`

6.9.1.2 `virtual
activemq::commands::ActiveMQBlobMessage::~ActiveMQBlobMessage ()
[inline, virtual]`

6.9.2 Member Function Documentation

6.9.2.1 `virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone
() const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 2168).

6.9.2.2 `virtual ActiveMQBlobMessage* ac-
tivismq::commands::ActiveMQBlobMessage::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2149).

6.9.2.3 `virtual void ac-
tivismq::commands::ActiveMQBlobMessage::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2150).

6.9.2.4 `virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 369).

6.9.2.5 `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new `DataStructure` (p. 1461) type copy.

Reimplemented from `activemq::commands::Message` (p. 2152).

6.9.2.6 `std::string activemq::commands::ActiveMQBlobMessage::getMimeType () const` [inline]

Get the Mime Type of the Blob.

Returns:

string holding the MIME Type.

6.9.2.7 `std::string activemq::commands::ActiveMQBlobMessage::getName () const` [inline]

Gets the Name of the Blob.

Returns:

string name of the Blob.

6.9.2.8 `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl () const` [inline]

Get the Remote URL of the Blob.

Returns:

string from of the Remote Blob URL.

6.9.2.9 `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker () const` [inline]

Gets if this Blob is deleted by the Broker.

Returns:

true if the Blob is deleted by the Broker.

6.9.2.10 `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool value)` [inline]

Sets the Deleted By Broker flag.

Parameters:

value - set the Delete by broker flag to value.

6.9.2.11 `void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & mimeType)` [inline]

Set the Mime Type of the Blob.

Parameters:

mimeType - String holding the MIME Type.

6.9.2.12 `void activemq::commands::ActiveMQBlobMessage::setName (const std::string & name)` [inline]

Sets the Name of the Blob.

Parameters:

name - Name of the Blob.

6.9.2.13 `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl (const std::string & remoteURL)` [inline]

Set the Remote URL of the Blob.

Parameters:

remoteURL - String form of the Remote URL.

6.9.2.14 `virtual std::string activemq::commands::ActiveMQBlobMessage::toString
() const [virtual]`

Returns a string containing the information for this **DataSet** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2159).

6.9.3 Field Documentation

6.9.3.1 `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_-
MIME_TYPE [static]`

6.9.3.2 `const unsigned char activemq::commands::ActiveMQBlobMessage::ID_-
ACTIVEMQBLOBMESSAGE = 29 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBlobMessage.h`

6.10 activemq:wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.177).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.10.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.177).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.10.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.10.3 Member Function Documentation

6.10.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.10.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.10.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.10.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.10.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2317).

6.10.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

6.10.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h`

6.11 activemq:wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 181).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual ~**ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.11.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.181).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.11.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.11.3 Member Function Documentation

6.11.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.11.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.11.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::marshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.11.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.11.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2327).

6.11.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

6.11.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h`

6.12 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 185).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.12.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 185).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.12.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.12.3 Member Function Documentation

6.12.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.12.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.12.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseMarshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.12.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.12.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2312).

6.12.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

6.12.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h`

6.13 activemq:wireformat::openwire::marshal:v5::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v5::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.13.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.13.3 Member Function Documentation

6.13.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.13.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.13.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.13.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.13.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2322).

6.13.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

6.13.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h`

6.14 activemq:wireformat::openwire::marshal:v2::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 193).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v2::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.14.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 193).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller()` [inline]

6.14.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller()` [inline, virtual]

6.14.3 Member Function Documentation

6.14.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.14.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.14.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.14.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.14.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2307).

6.14.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

6.14.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h`

6.15 activemq::commands::ActiveMQBytesMessage Class Reference

#include <src/main/activemq/commands/ActiveMQBytesMessage.h> Inheritance diagram for activemq::commands::ActiveMQBytesMessage:

Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQBytesMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **onSend** ()
Store the Data that was written to the stream before a send.
- virtual void **setBodyBytes** (const unsigned char *buffer, std::size_t numBytes) throw (cms::MessageNotWriteableException, cms::CMSException)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const throw (cms::MessageNotReadableException, cms::CMSException)
Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.
- virtual std::size_t **getBodyLength** () const throw (cms::MessageNotReadableException, cms::CMSException)
Returns the number of bytes contained in the body of this message.

- virtual void **reset** () throw (cms::MessageFormatException, cms::CMSEException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the bytes message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the bytes message stream.
- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Char from the Bytes message stream.
- virtual void **writeChar** (char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a char to the bytes message stream as a 1-byte value.
- virtual float **readFloat** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a float to the bytes message stream as a 4 byte value.

- virtual double **readDouble** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Bytes message stream.

- virtual void **writeInt** (int value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Bytes message stream.

- virtual std::string **readUTF** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads an UTF String from the BytesMessage stream.
- virtual void **writeUTF** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an UTF String to the BytesMessage stream.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQBYTESMESSAGE** = 24

6.15.1 Constructor & Destructor Documentation

6.15.1.1 `activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ()`

6.15.1.2 `virtual
activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage
() [virtual]`

6.15.2 Member Function Documentation

6.15.2.1 `virtual void activemq::commands::ActiveMQBytesMessage::clearBody ()
throw (cms::CMSEException) [virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 368).

6.15.2.2 `virtual cms::BytesMessage* ac-
tivemq::commands::ActiveMQBytesMessage::clone () const
[inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements `cms::BytesMessage` (p. 941).

6.15.2.3 `virtual ActiveMQBytesMessage* ac-
tivemq::commands::ActiveMQBytesMessage::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2149).

6.15.2.4 `virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2150).

6.15.2.5 `virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Parameters:

value The `Command` (p. 1063) to compare to this one.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 369).

6.15.2.6 `virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes () const throw (cms::MessageNotReadableException, cms::CMSException) [virtual]`

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer. Call `getBodyLength` to determine the number of bytes to expect.

Returns:

const pointer to a byte buffer

Exceptions:

CMSException - If an internal error occurs.

MessageNotReadableException - If the message is in Write Only Mode.

Implements `cms::BytesMessage` (p. 941).

6.15.2.7 `virtual std::size_t activemq::commands::ActiveMQBytesMessage::getBodyLength () const throw (cms::MessageNotReadableException, cms::CMSException) [virtual]`

Returns the number of bytes contained in the body of this message.

Returns:

number of bytes.

Exceptions:

CMSException - If an internal error occurs.

MessageNotReadableException - If the message is in Write Only Mode.

Implements `cms::BytesMessage` (p. 942).

6.15.2.8 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new `DataStructure` (p. 1461) type copy.

Reimplemented from `activemq::commands::Message` (p. 2152).

6.15.2.9 `virtual void activemq::commands::ActiveMQBytesMessage::onSend () [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 376).

6.15.2.10 `virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Boolean from the Bytes message stream.

Returns:

boolean value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 942).

6.15.2.11 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte() const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Byte from the Bytes message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 942).

6.15.2.12 `virtual std::size_t activemq::commands::ActiveMQBytesMessage::readBytes(unsigned char * buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 943).

6.15.2.13 `virtual std::size_t activemq::commands::ActiveMQBytesMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 943).

6.15.2.14 `virtual char activemq::commands::ActiveMQBytesMessage::readChar () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Char from the Bytes message stream.

Returns:

char value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 944).

6.15.2.15 `virtual double activemq::commands::ActiveMQBytesMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a 64 bit double from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 944).

6.15.2.16 virtual float activemq::commands::ActiveMQBytesMessage::readFloat
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 32 bit float from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 945).

6.15.2.17 virtual int activemq::commands::ActiveMQBytesMessage::readInt
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns:

int value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 945).

6.15.2.18 virtual long long ac-
tivismq::commands::ActiveMQBytesMessage::readLong
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 64 bit long from the Bytes message stream.

Returns:

long long value from stream

Exceptions:

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 945).

6.15.2.19 virtual short activemq::commands::ActiveMQBytesMessage::readShort
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns:

short value from stream

Exceptions:

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 946).

6.15.2.20 virtual std::string ac-
tivismq::commands::ActiveMQBytesMessage::readString
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads an ASCII String from the Bytes message stream.

Returns:

String from stream

Exceptions:

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 946).

6.15.2.21 virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements cms::BytesMessage (p. 946).

6.15.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads an UTF String from the BytesMessage stream.

Returns:

String from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements cms::BytesMessage (p. 947).

6.15.2.23 virtual void activemq::commands::ActiveMQBytesMessage::reset ()
throw (cms::MessageFormatException, cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions:

CMSException - If the provider fails to perform the reset operation.

MessageFormatException - If the Message (p. 2145) has an invalid format.

Implements cms::BytesMessage (p. 947).

6.15.2.24 `virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * buffer, std::size_t numBytes) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

sets the bytes given to the message body.

Parameters:

buffer Byte Buffer to copy

numBytes Number of bytes in Buffer to copy

Exceptions:

CMSEException - If an internal error occurs.

MessageNotWriteableException - if in Read Only Mode.

Implements `cms::BytesMessage` (p. 947).

6.15.2.25 `virtual std::string activemq::commands::ActiveMQBytesMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2159).

6.15.2.26 `virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value boolean to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements `cms::BytesMessage` (p. 948).

6.15.2.27 `virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters:

value byte to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 948).

6.15.2.28 `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes
(const unsigned char * value, std::size_t offset, std::size_t length) throw
(cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

Parameters:

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 948).

6.15.2.29 `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes
(const std::vector< unsigned char > & value) throw (
cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters:

value bytes to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 949).

6.15.2.30 `virtual void activemq::commands::ActiveMQBytesMessage::writeChar
(char value) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]`

Writes a char to the bytes message stream as a 1-byte value.

Parameters:

value char to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 949).

6.15.2.31 virtual void activemq::commands::ActiveMQBytesMessage::writeDouble
(double *value*) throw (cms::MessageNotWriteableException,
cms::CMSEException) [virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters:

value double to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 949).

6.15.2.32 virtual void activemq::commands::ActiveMQBytesMessage::writeFloat
(float *value*) throw (cms::MessageNotWriteableException,
cms::CMSEException) [virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters:

value float to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 950).

6.15.2.33 virtual void activemq::commands::ActiveMQBytesMessage::writeInt (int
value) throw (cms::MessageNotWriteableException, cms::CMSEException
) [virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters:

value signed int to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 950).

6.15.2.34 virtual void activemq::commands::ActiveMQBytesMessage::writeLong
(long long *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters:

value signed long long to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 950).

6.15.2.35 virtual void activemq::commands::ActiveMQBytesMessage::writeShort
(short *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters:

value signed short to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 951).

6.15.2.36 virtual void activemq::commands::ActiveMQBytesMessage::writeString
(const std::string & *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes an ASCII String to the Bytes message stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 951).

6.15.2.37 virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value unsigned short to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements cms::BytesMessage (p. 951).

6.15.2.38 virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes an UTF String to the BytesMessage stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

Implements cms::BytesMessage (p. 952).

6.15.3 Field Documentation

6.15.3.1 const unsigned char activemq::commands::ActiveMQBytesMessage::ID_ - ACTIVEMQBYTESMESSAGE = 24 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQBytesMessage.h**

6.16

`activemq:wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`

Class Reference

6.16 `activemq:wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 213).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller:
```

Public Member Functions

- `ActiveMQBytesMessageMarshaller ()`
- `virtual ~ActiveMQBytesMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.16.1 Detailed Description

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 213).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.16.2 Constructor & Destructor Documentation

6.16.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

6.16.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

6.16.3 Member Function Documentation

6.16.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.16.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.16.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.16

activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

Class Reference

217

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2316).

6.16.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2316).

6.16.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2317).

6.16.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

6.16.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h`

6.17

activemq:wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

Class Reference

6.17 ~~activemq:wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller~~²¹⁹

Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller:
```

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.17.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

6.17.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

6.17.3 Member Function Documentation

6.17.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.17.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.17.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.17

activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

Class Reference

221

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2326).

6.17.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2326).

6.17.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2327).

6.17.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

6.17.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h`

6.18

activemq:wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

Class Reference

6.18 ~~activemq:wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller~~²²³

Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller:
```

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.18.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.18.2 Constructor & Destructor Documentation

6.18.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

6.18.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

6.18.3 Member Function Documentation

6.18.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.18.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.18.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.18

activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

Class Reference

225

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2311).

6.18.3.4 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2311).

6.18.3.5 virtual int ac-

```
tivemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2312).

6.18.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

6.18.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h`

6.19

`activemq:wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`

Class Reference

~~6.19~~ `activemq:wireformat::openwire::marshal::v5::ActiveMQBytesMessage` ²²⁷

Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).

`#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h>` Inherits from `activemq:wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.19.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.19.2 Constructor & Destructor Documentation

6.19.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

6.19.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

6.19.3 Member Function Documentation

6.19.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.19.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.19.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.19

activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller

Class Reference

229

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2321).

6.19.3.4 virtual void ac-

```
timemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2321).

6.19.3.5 virtual int ac-

```
timemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2322).

6.19.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

6.19.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h`

6.20

activemq:wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

Class Reference

6.20 ~~activemq:wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller~~²³¹

Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller:
```

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.20.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.20.2 Constructor & Destructor Documentation

6.20.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller()` [inline]

6.20.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller()` [inline, virtual]

6.20.3 Member Function Documentation

6.20.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.20.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.20.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.20

activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

Class Reference

233

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2306).

6.20.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2306).

6.20.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2307).

6.20.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

6.20.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h`

6.21 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

#include <src/main/activemq/core/ActiveMQConnection.h> Inheritance diagram for activemq::core::ActiveMQConnection:

Public Member Functions

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)
Constructor.
- virtual ~**ActiveMQConnection** ()
- virtual void **removeSession** (**ActiveMQSession** *session) throw (cms::CMSException)
Removes the session resources for the given session instance.
- virtual void **addProducer** (**ActiveMQProducer** *producer) throw (cms::CMSException)
Adds an active Producer to the Set of known producers.
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw (cms::CMSException)
Removes an active Producer to the Set of known producers.
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** *dispatcher) throw (cms::CMSException)
Adds a dispatcher for a consumer.
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer) throw (cms::CMSException)
Removes the dispatcher for a consumer.
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** *consumer, long long timeout) throw (exceptions::ActiveMQException)
If supported sends a message pull request to the service provider asking for the delivery of a new message.
- bool **isClosed** () const
Checks if this connection has been closed.
- bool **isStarted** () const
Check if this connection has been started.
- virtual void **destroyDestination** (const **commands::ActiveMQDestination** *destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)

Requests that the Broker removes the given Destination.

- virtual void **destroyDestination** (const **cms::Destination** *destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)

Requests that the Broker removes the given Destination.

- virtual const **cms::ConnectionMetaData** * **getMetaData** () const throw (cms::CMSException)

Gets the metadata for this connection.

- virtual **cms::Session** * **createSession** () throw (cms::CMSException)

Creates a new Session to work for this Connection.

- virtual **cms::Session** * **createSession** (**cms::Session::AcknowledgeMode** ackMode) throw (cms::CMSException)

Creates a new Session to work for this Connection using the specified acknowledgment mode.

- virtual std::string **getClientID** () const

Get the Client Id for this session.

- virtual void **close** () throw (cms::CMSException)

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

- virtual void **start** () throw (cms::CMSException)

Starts or (restarts) a connections delivery of incoming messages.

- virtual void **stop** () throw (cms::CMSException)

Stop the flow of incoming messages.

- virtual **cms::ExceptionListener** * **getExceptionListener** () const

Gets the registered Exception Listener for this connection.

- virtual void **setExceptionListener** (**cms::ExceptionListener** *listener)

Sets the registered Exception Listener for this connection.

- void **addTransportListener** (**transport::TransportListener** *transportListener)

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

- void **removeTransportListener** (**transport::TransportListener** *transportListener)

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.

- virtual void **onCommand** (const **Pointer< commands::Command >** &command)

Event handler for the receipt of a non-response command from the transport.

- virtual void **onException** (const **decaf::lang::Exception** &ex)

Event handler for an exception from a command transport.

- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- const **commands::ConnectionInfo** & **getConnectionInfo** () const throw (exceptions::ActiveMQException)
Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.
- const **commands::ConnectionId** & **getConnectionId** () const throw (exceptions::ActiveMQException)
Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.
- void **oneway** (Pointer< **commands::Command** > command) throw (activemq::exceptions::ActiveMQException)
Sends a oneway message.
- void **syncRequest** (Pointer< **commands::Command** > command, unsigned int timeout=0) throw (activemq::exceptions::ActiveMQException)
Sends a synchronous request and returns the response from the broker.
- virtual void **fire** (const exceptions::ActiveMQException &ex)
Notify the exception listener.

6.21.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 activemq::core::ActiveMQConnection::ActiveMQConnection (const Pointer< transport::Transport > & *transport*, const Pointer< decaf::util::Properties > & *properties*)

Constructor.

Parameters:

transport The Transport requested for this connection to the Broker.

properties The Properties that were defined for this connection

6.21.2.2 `virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection ()`
[virtual]

6.21.3 Member Function Documentation

6.21.3.1 `virtual void activemq::core::ActiveMQConnection::addDispatcher`
(const Pointer< commands::ConsumerId > & *consumer*, Dispatcher *
dispatcher) throw (cms::CMSException) [virtual]

Adds a dispatcher for a consumer.

Parameters:

consumer - The consumer for which to register a dispatcher.

dispatcher - The dispatcher to handle incoming messages for the consumer.

6.21.3.2 `virtual void activemq::core::ActiveMQConnection::addProducer`
(ActiveMQProducer * *producer*) throw (cms::CMSException) [virtual]

Adds an active Producer to the Set of known producers.

Parameters:

producer - The Producer to add from the the known set.

6.21.3.3 `void activemq::core::ActiveMQConnection::addTransportListener`
(transport::TransportListener * *transportListener*)

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class. Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

Parameters:

transportListener The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.

6.21.3.4 `virtual void activemq::core::ActiveMQConnection::close () throw (`
`cms::CMSException)` [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions:

CMSException

Implements `cms::Connection` (p. 1132).

6.21.3.5 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode ackMode) throw (cms::CMSException) [virtual]`

Creates a new Session to work for this Connection using the specified acknowledgment mode.

Parameters:

ackMode the Acknowledgment Mode to use.

Exceptions:

CMSException

Implements `cms::Connection` (p. 1132).

6.21.3.6 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession () throw (cms::CMSException) [virtual]`

Creates a new Session to work for this Connection.

Exceptions:

CMSException

Implements `cms::Connection` (p. 1133).

6.21.3.7 `virtual void activemq::core::ActiveMQConnection::destroyDestination (const cms::Destination * destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]`

Requests that the Broker removes the given Destination. Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters:

destination The CMS Destination the Broker will be requested to remove.

Exceptions:

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.21.3.8 `virtual void activemq::core::ActiveMQConnection::destroyDestination (const commands::ActiveMQDestination * destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]`

Requests that the Broker removes the given Destination. Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters:

destination The Destination the Broker will be requested to remove.

Exceptions:

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.21.3.9 `virtual void activemq::core::ActiveMQConnection::fire (const exceptions::ActiveMQException & ex) [virtual]`

Notify the exception listener.

Parameters:

ex the exception to fire

6.21.3.10 `virtual std::string activemq::core::ActiveMQConnection::getClientID () const [virtual]`

Get the Client Id for this session.

Returns:

string version of Client Id

Implements **cms::Connection** (p. 1133).

6.21.3.11 `const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId () const throw (exceptions::ActiveMQException)`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

6.21.3.12 `const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo () const throw (exceptions::ActiveMQException)`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

6.21.3.13 `virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener () const [inline, virtual]`

Gets the registered Exception Listener for this connection.

Returns:

pointer to an exception listener or NULL

Implements **cms::Connection** (p. 1133).

6.21.3.14 `virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData () const throw (cms::CMSException) [inline, virtual]`

Gets the metadata for this connection.

Returns:

the connection MetaData pointer (caller does not own it).

Exceptions:

CMSException if the provider fails to get the connection metadata for this connection.

See also:

ConnectionMetaData

Since:

2.0

Implements **cms::Connection** (p. 1133).

6.21.3.15 `bool activemq::core::ActiveMQConnection::isClosed () const [inline]`

Checks if this connection has been closed.

Returns:

true if the connection is closed

6.21.3.16 `bool activemq::core::ActiveMQConnection::isStarted () const [inline]`

Check if this connection has been started.

Returns:

true if the start method has been called.

6.21.3.17 `virtual void activemq::core::ActiveMQConnection::onCommand (const Pointer< commands::Command > & command) [virtual]`

Event handler for the receipt of a non-response command from the transport.

Parameters:

command the received command object.

Implements `activemq::transport::TransportListener` (p. 3289).

6.21.3.18 `void activemq::core::ActiveMQConnection::oneway (Pointer< commands::Command > command) throw (activemq::exceptions::ActiveMQException)`

Sends a oneway message.

Parameters:

command The message to send.

Exceptions:

ConnectorException if not currently connected, or if the operation fails for any reason.

6.21.3.19 `virtual void activemq::core::ActiveMQConnection::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters:

ex The exception.

Implements `activemq::transport::TransportListener` (p. 3290).

6.21.3.20 `virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & consumer) throw (cms::CMSException) [virtual]`

Removes the dispatcher for a consumer.

Parameters:

consumer - The consumer for which to remove the dispatcher.

6.21.3.21 `virtual void activemq::core::ActiveMQConnection::removeProducer (const Pointer< commands::ProducerId > & producerId) throw (cms::CMSException) [virtual]`

Removes an active Producer to the Set of known producers.

Parameters:

producerId - The ProducerId to remove from the the known set.

6.21.3.22 `virtual void activemq::core::ActiveMQConnection::removeSession (ActiveMQSession * session) throw (cms::CMSException) [virtual]`

Removes the session resources for the given session instance.

Parameters:

session The session to be unregistered from this connection.

6.21.3.23 `void activemq::core::ActiveMQConnection::removeTransportListener (transport::TransportListener * transportListener)`

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events. The caller is responsible for freeing the listener in all cases.

Parameters:

transportListener The pointer to the TransportListener to remove from the set of listeners.

6.21.3.24 `virtual void activemq::core::ActiveMQConnection::sendPullRequest (const commands::ConsumerInfo * consumer, long long timeout) throw (exceptions::ActiveMQException) [virtual]`

If supported sends a message pull request to the service provider asking for the delivery of a new message. This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

Parameters:

consumer - the ConsumerInfo for the requesting Consumer.

timeout - the time that the client is willing to wait.

6.21.3.25 `virtual void activemq::core::ActiveMQConnection::setExceptionListener (cms::ExceptionListener * listener) [inline, virtual]`

Sets the registered Exception Listener for this connection.

Parameters:

listener pointer to and ExceptionListener

Implements `cms::Connection` (p. 1134).

6.21.3.26 `virtual void activemq::core::ActiveMQConnection::start () throw (cms::CMSException) [virtual]`

Starts or (restarts) a connections delivery of incoming messages.

Exceptions:

CMSException

Implements `cms::Startable` (p. 3014).

6.21.3.27 `virtual void activemq::core::ActiveMQConnection::stop () throw (cms::CMSException) [virtual]`

Stop the flow of incoming messages.

Exceptions:

CMSException

Implements `cms::Stoppable` (p. 3076).

6.21.3.28 `void activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw (activemq::exceptions::ActiveMQException)`

Sends a synchronous request and returns the response from the broker. Converts any error responses into an exception.

Parameters:

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions:

ConnectorException thrown if an error response was received from the broker, or if any other error occurred.

6.21.3.29 `virtual void activemq::core::ActiveMQConnection::transportInterrupted () [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3290).

6.21.3.30 `virtual void activemq::core::ActiveMQConnection::transportResumed () [virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3290).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnection.h`

6.22 activemq::core::ActiveMQConnectionFactory Class Reference

#include <src/main/activemq/core/ActiveMQConnectionFactory.h> Inheritance diagram for activemq::core::ActiveMQConnectionFactory:

Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")
Constructor.
- virtual ~**ActiveMQConnectionFactory** ()
- virtual **cms::Connection * createConnection** () throw (cms::CMSEException)
Creates a connection with the default user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password) throw (cms::CMSEException)
Creates a connection with the specified user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw (cms::CMSEException)
Creates a connection with the specified user identity.
- virtual void **setUsername** (const std::string &username)
Sets the username that should be used when creating a new connection.
- virtual const std::string & **getUsername** () const
Gets the username that this factory will use when creating a new connection instance.
- virtual void **setPassword** (const std::string &password)
Sets the password that should be used when creating a new connection.
- virtual const std::string & **getPassword** () const
Gets the password that this factory will use when creating a new connection instance.
- virtual void **setBrokerURL** (const std::string &brokerURL)
Sets the Broker URL that should be used when creating a new connection instance.
- virtual const std::string & **getBrokerURL** () const
Gets the Broker URL that this factory will use when creating a new connection instance.

Static Public Member Functions

- static **cms::Connection * createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw (cms::CMSEException)

Creates a connection with the specified user identity.

6.22.1 Constructor & Destructor Documentation

6.22.1.1 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory**
()

6.22.1.2 **activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory**
(const std::string & url, const std::string & username = "", const
std::string & password = "")

Constructor.

Parameters:

url the URL of the Broker we are connecting to.

username to authenticate with, defaults to ""

password to authenticate with, defaults to ""

6.22.1.3 **virtual**
activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory
() [inline, virtual]

6.22.2 Member Function Documentation

6.22.2.1 **static cms::Connection* ac-**
tivemq::core::ActiveMQConnectionFactory::createConnection (const
std::string & url, const std::string & username, const std::string &
password, const std::string & *clientId* = "") throw (cms::CMSEException
) [static]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Parameters:

url the URL of the Broker we are connecting to.

username to authenticate with

password to authenticate with

clientId to assign to connection, defaults to ""

Exceptions:

CMSEException.

6.22.2.2 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) throw (cms::CMSEException) [virtual]`

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username to authenticate with

password to authenticate with

clientId to assign to connection if "" then a random client Id is created for this connection.

Returns:

a Connection Pointer

Exceptions:

CMSEException

Implements `cms::ConnectionFactory` (p. 1185).

6.22.2.3 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password) throw (cms::CMSEException) [virtual]`

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username to authenticate with

password to authenticate with

Returns:

a Connection Pointer

Exceptions:

CMSEException

Implements `cms::ConnectionFactory` (p. 1186).

6.22.2.4 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection () throw (cms::CMSEException) [virtual]`

Creates a connection with the default user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Returns:

a Connection Pointer

Exceptions:

CMSException

Implements **cms::ConnectionFactory** (p. 1186).

6.22.2.5 `virtual const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL () const [inline, virtual]`

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns:

brokerURL string

6.22.2.6 `virtual const std::string& activemq::core::ActiveMQConnectionFactory::getPassword () const [inline, virtual]`

Gets the password that this factory will use when creating a new connection instance.

Returns:

password string, "" for default credentials

6.22.2.7 `virtual const std::string& activemq::core::ActiveMQConnectionFactory::getUsername () const [inline, virtual]`

Gets the username that this factory will use when creating a new connection instance.

Returns:

username string, "" for default credentials

6.22.2.8 `virtual void activemq::core::ActiveMQConnectionFactory::setBrokerURL (const std::string & brokerURL) [inline, virtual]`

Sets the Broker URL that should be used when creating a new connection instance.

Parameters:

brokerURL string

6.22.2.9 `virtual void activemq::core::ActiveMQConnectionFactory::setPassword
(const std::string & password)` [inline, virtual]

Sets the password that should be used when creating a new connection.

Parameters:

password string

6.22.2.10 `virtual void activemq::core::ActiveMQConnectionFactory::setUsername
(const std::string & username)` [inline, virtual]

Sets the username that should be used when creating a new connection.

Parameters:

username string

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionFactory.h`

6.23 activemq::core::ActiveMQConnectionMetaData Class Reference

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.

#include <src/main/activemq/core/ActiveMQConnectionMetaData.h> Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

Public Member Functions

- **ActiveMQConnectionMetaData** ()
- virtual **~ActiveMQConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const throw (cms::CMSEException)
Gets the CMS API version.
- virtual int **getCMSMajorVersion** () const throw (cms::CMSEException)
Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () const throw (cms::CMSEException)
Gets the CMS minor version number.
- virtual std::string **getCMSProviderName** () const throw (cms::CMSEException)
Gets the CMS provider name.
- virtual std::string **getProviderVersion** () const throw (cms::CMSEException)
Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () const throw (cms::CMSEException)
Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () const throw (cms::CMSEException)
Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () const throw (cms::CMSEException)
Gets an Vector of the CMSX property names.

6.23.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.

Since:

3.0

6.23.2 Constructor & Destructor Documentation

6.23.2.1 `activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData()`

6.23.2.2 `virtual activemq::core::ActiveMQConnectionMetaData::~~ActiveMQConnectionMetaData()` [virtual]

6.23.3 Member Function Documentation

6.23.3.1 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion()`
`const throw (cms::CMSException)` [virtual]

Gets the CMS major version number.

Returns:

the CMS API major version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1238).

6.23.3.2 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion()`
`const throw (cms::CMSException)` [virtual]

Gets the CMS minor version number.

Returns:

the CMS API minor version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1238).

6.23.3.3 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName()`
`const throw (cms::CMSException)` [virtual]

Gets the CMS provider name.

Returns:

the CMS provider name

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1238).

6.23.3.4 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS API version.

Returns:

the CMS API Version in String form.

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1239).

6.23.3.5 `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames () const throw (cms::CMSException) [virtual]`

Gets an Vector of the CMSX property names.

Returns:

an Vector of CMSX property names

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1239).

6.23.3.6 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider major version number.

Returns:

the CMS provider major version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1239).

6.23.3.7 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider minor version number.

Returns:

the CMS provider minor version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1239).

6.23.3.8 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider version.

Returns:

the CMS provider version

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p.1240).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionMetaData.h`

6.24 activemq::core::ActiveMQConnectionSupport Class Reference

#include <src/main/activemq/core/ActiveMQConnectionSupport.h> Inheritance diagram for activemq::core::ActiveMQConnectionSupport:

Public Member Functions

- **ActiveMQConnectionSupport** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)
*Creates an instance of the **ActiveMQConnectionSupport** (p. 253) class, the most common properties for a connection are pulled from the properties instance or are set to defaults.*
- virtual ~**ActiveMQConnectionSupport** ()
- virtual void **startupTransport** () throw (decaf::lang::Exception)
Starts the Transport, this should initiate the connection between this client and the Transports endpoint.
- virtual void **shutdownTransport** () throw (decaf::lang::Exception)
Closes this object and deallocates the appropriate resources.
- const **decaf::util::Properties** & **getProperties** () const
Gets the Properties object that this Config object was initialized with.
- **transport::Transport** & **getTransport** () const
Gets the Transport Configured for this Connection.
- bool **isAlwaysSyncSend** () const
Gets if the Connection should always send things Synchronously.
- void **setAlwaysSyncSend** (bool value)
Sets if the Connection should always send things Synchronously.
- bool **isUseAsyncSend** () const
Gets if the useAsyncSend option is set.
- void **setUseAsyncSend** (bool value)
Sets the useAsyncSend option.
- unsigned int **getSendTimeout** () const
Gets the assigned send timeout for this Connector.
- void **setSendTimeout** (unsigned int timeout)
Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.
- unsigned int **getCloseTimeout** () const
Gets the assigned close timeout for this Connector.

- void **setCloseTimeout** (unsigned int timeout)
Sets the close timeout to use when sending the disconnect request.
- unsigned int **getProducerWindowSize** () const
Gets the configured producer window size for Producers that are created from this connector.
- void **setProducerWindowSize** (unsigned int windowSize)
Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.
- std::string **getUsername** () const
Gets the Configured Username.
- void **setUsername** (const std::string &username)
Sets the Username.
- std::string **getPassword** () const
Gets the Configured Password.
- void **setPassword** (const std::string &password)
Sets the Password.
- std::string **getClientId** () const
Gets the Configured Client Id.
- void **setClientId** (const std::string &clientId)
Sets the Client Id.
- long long **getNextSessionId** ()
Get the Next available Session Id.
- long long **getNextTempDestinationId** ()
Get the Next Temporary Destination Id.
- long long **getNextLocalTransactionId** ()
Get the Next Temporary Destination Id.

6.24.1 Constructor & Destructor Documentation

6.24.1.1 **activemq::core::ActiveMQConnectionSupport::ActiveMQConnectionSupport** (const Pointer< transport::Transport > & *transport*, const Pointer< decaf::util::Properties > & *properties*)

Creates an instance of the **ActiveMQConnectionSupport** (p. 253) class, the most common properties for a connection are pulled from the properties instance or are set to defaults.

Parameters:

- transport*** The Transport that this Connection will use for sending Commands to the Broker.
- properties*** The URI configured properties for this connection.

6.24.1.2 `virtual
activemq::core::ActiveMQConnectionSupport::~~ActiveMQConnectionSupport
() [virtual]`

6.24.2 Member Function Documentation

6.24.2.1 `std::string activemq::core::ActiveMQConnectionSupport::getClientId ()
const [inline]`

Gets the Configured Client Id.

Returns:

the clientId.

6.24.2.2 `unsigned int ac-
tivemq::core::ActiveMQConnectionSupport::getCloseTimeout () const
[inline]`

Gets the assigned close timeout for this Connector.

Returns:

the close timeout configured in the connection uri

6.24.2.3 `long long ac-
tivemq::core::ActiveMQConnectionSupport::getNextLocalTransactionId ()
[inline]`

Get the Next Temporary Destination Id.

Returns:

the next id in the sequence.

6.24.2.4 `long long activemq::core::ActiveMQConnectionSupport::getNextSessionId
() [inline]`

Get the Next available Session Id.

Returns:

the next id in the sequence.

6.24.2.5 `long long ac-
tivemq::core::ActiveMQConnectionSupport::getNextTempDestinationId ()
[inline]`

Get the Next Temporary Destination Id.

Returns:

the next id in the sequence.

6.24.2.6 `std::string activemq::core::ActiveMQConnectionSupport::getPassword ()`
`const [inline]`

Gets the Configured Password.

Returns:

the password.

6.24.2.7 `unsigned int activemq::core::ActiveMQConnectionSupport::getProducerWindowSize ()`
`const [inline]`

Gets the configured producer window size for Producers that are created from this connector. This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns:

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.24.2.8 `const decaf::util::Properties& activemq::core::ActiveMQConnectionSupport::getProperties ()`
`const [inline]`

Gets the Properties object that this Config object was initialized with.

Returns:

a const reference to the Connection Config.

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

6.24.2.9 `unsigned int activemq::core::ActiveMQConnectionSupport::getSendTimeout ()`
`const [inline]`

Gets the assigned send timeout for this Connector.

Returns:

the send timeout configured in the connection uri

6.24.2.10 `transport::Transport& activemq::core::ActiveMQConnectionSupport::getTransport ()`
`const [inline]`

Gets the Transport Configured for this Connection.

Returns:

the configured transport

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

6.24.2.11 `std::string activemq::core::ActiveMQConnectionSupport::getUsername ()`
`const [inline]`

Gets the Configured Username.

Returns:

the username.

6.24.2.12 `bool activemq::core::ActiveMQConnectionSupport::isAlwaysSyncSend ()`
`const [inline]`

Gets if the Connection should always send things Synchronously.

Returns:

true if sends should always be Synchronous.

6.24.2.13 `bool activemq::core::ActiveMQConnectionSupport::isUseAsyncSend ()`
`const [inline]`

Gets if the useAsyncSend option is set.

Returns:

true if on false if not.

6.24.2.14 `void activemq::core::ActiveMQConnectionSupport::setAlwaysSyncSend`
`(bool value) [inline]`

Sets if the Connection should always send things Synchronously.

Parameters:

value true if sends should always be Synchronous.

6.24.2.15 `void activemq::core::ActiveMQConnectionSupport::setClientId (const`
`std::string & clientId) [inline]`

Sets the Client Id.

Parameters:

clientId - The new clientId value.

6.24.2.16 `void activemq::core::ActiveMQConnectionSupport::setCloseTimeout`
`(unsigned int timeout) [inline]`

Sets the close timeout to use when sending the disconnect request.

Parameters:

timeout - The time to wait for a close message.

6.24.2.17 `void activemq::core::ActiveMQConnectionSupport::setPassword (const std::string & password) [inline]`

Sets the Password.

Parameters:

password - The new password value.

6.24.2.18 `void activemq::core::ActiveMQConnectionSupport::setProducerWindowSize (unsigned int windowSize) [inline]`

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters:

windowSize - The size in bytes of the Producers memory window.

6.24.2.19 `void activemq::core::ActiveMQConnectionSupport::setSendTimeout (unsigned int timeout) [inline]`

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters:

timeout - The time to wait for a response.

6.24.2.20 `void activemq::core::ActiveMQConnectionSupport::setUseAsyncSend (bool value) [inline]`

Sets the useAsyncSend option.

Parameters:

value - true to activate, false to disable.

6.24.2.21 `void activemq::core::ActiveMQConnectionSupport::setUsername (const std::string & username) [inline]`

Sets the Username.

Parameters:

username - The new username value.

6.24.2.22 virtual void activemq::core::ActiveMQConnectionSupport::shutdownTransport () throw (decaf::lang::Exception) [virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

Exception

6.24.2.23 virtual void activemq::core::ActiveMQConnectionSupport::startupTransport () throw (decaf::lang::Exception) [virtual]

Starts the Transport, this should initiate the connection between this client and the Transports endpoint.

Exceptions:

Exception

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConnectionSupport.h

6.25 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Data Structures

- class `StaticInitializer`

Public Types

- enum `TransactionState` {
`TRANSACTION_STATE_BEGIN` = 0, `TRANSACTION_STATE_PREPARE` = 1, `TRANSACTION_STATE_COMMITONEPHASE` = 2, `TRANSACTION_STATE_COMMITTWOPHASE` = 3,
`TRANSACTION_STATE_ROLLBACK` = 4, `TRANSACTION_STATE_RECOVER` = 5, `TRANSACTION_STATE_FORGET` = 6, `TRANSACTION_STATE_END` = 7 }
 - enum `DestinationActions` { `DESTINATION_ADD_OPERATION` = 0, `DESTINATION_REMOVE_OPERATION` = 1 }
 - enum `AckType` {
`ACK_TYPE_DELIVERED` = 0, `ACK_TYPE_POISON` = 1, `ACK_TYPE_CONSUMED` = 2, `ACK_TYPE_REDELIVERED` = 3,
`ACK_TYPE_INDIVIDUAL` = 4 }
 - enum `DestinationOption` {
`CONSUMER_PREFETCHSIZE`, `CONSUMER_MAXPENDINGMSGLIMIT`, `CONSUMER_NOLOCAL`, `CONSUMER_DISPATCHASYNC`,
`CONSUMER_RETROACTIVE`, `CONSUMER_SELECTOR`, `CONSUMER_EXCLUSIVE`, `CONSUMER_PRIORITY`,
`NUM_OPTIONS` }
- These values represent the options that can be appended to an Destination name, i.e.*
- enum `URIParam` {
`CONNECTION_SENDTIMEOUT`, `CONNECTION_PRODUCERWINDOWSIZE`, `CONNECTION_CLOSETIMEOUT`,
`CONNECTION_ALWAYSSENDSYNC`, `CONNECTION_USEASYNCSYNC`, `PARAM_USERNAME`, `PARAM_PASSWORD`, `PARAM_CLIENTID`,
`NUM_PARAMS` }
- These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.*

Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption** **toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam** **toURIOption** (const std::string &option)

6.25.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

```
ACK_TYPE_DELIVERED
ACK_TYPE_POISON
ACK_TYPE_CONSUMED
ACK_TYPE_REDELIVERED
ACK_TYPE_INDIVIDUAL
```

6.25.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

```
DESTINATION_ADD_OPERATION
DESTINATION_REMOVE_OPERATION
```

6.25.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e. /topic/-foo?consumer.exclusive=true

Enumerator:

```
CONSUMER_PREFETCHSIZE
CUNSUMER_MAXPENDINGMSGLIMIT
CONSUMER_NOLOCAL
CONSUMER_DISPATCHASYNC
CONSUMER_RETROACTIVE
CONSUMER_SELECTOR
CONSUMER_EXCLUSIVE
CONSUMER_PRIORITY
NUM_OPTIONS
```

6.25.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

```
TRANSACTION_STATE_BEGIN
TRANSACTION_STATE_PREPARE
TRANSACTION_STATE_COMMITONEPHASE
TRANSACTION_STATE_COMMITTWOPHASE
TRANSACTION_STATE_ROLLBACK
TRANSACTION_STATE_RECOVER
TRANSACTION_STATE_FORGET
TRANSACTION_STATE_END
```

6.25.2.5 enum activemq::core::ActiveMQConstants::URIParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

```
CONNECTION_SENDDTIMEOUT
CONNECTION_PRODUCERWINDOWSIZE
CONNECTION_CLOSETIMEOUT
CONNECTION_ALWAYSASYNCSEND
CONNECTION_USEASYNCSEND
PARAM_USERNAME
PARAM_PASSWORD
PARAM_CLIENTID
NUM_PARAMS
```

6.25.3 Member Function Documentation

- 6.25.3.1 static DestinationOption activemq::core::ActiveMQConstants::toDestinationOption (const std::string & *option*) [inline, static]
- 6.25.3.2 static const std::string& activemq::core::ActiveMQConstants::toString (const URIParam *option*) [inline, static]
- 6.25.3.3 static const std::string& activemq::core::ActiveMQConstants::toString (const DestinationOption *option*) [inline, static]
- 6.25.3.4 static URIParam activemq::core::ActiveMQConstants::toURIOption (const std::string & *option*) [inline, static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConstants.h

6.26 activemq::core::ActiveMQConsumer Class Reference

#include <src/main/activemq/core/ActiveMQConsumer.h> Inheritance diagram for activemq::core::ActiveMQConsumer:

Public Member Functions

- **ActiveMQConsumer** (const **Pointer**< **commands::ConsumerInfo** > &consumerInfo, **ActiveMQSession** *session, const **Pointer**< **ActiveMQTransactionContext** > &transaction)
Constructor.
- virtual ~**ActiveMQConsumer** ()
- virtual void **start** ()
Starts the Consumer if not already started and not closed.
- virtual void **stop** ()
Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.
- virtual void **close** () throw (cms::CMSException)
Closes the Consumer.
- virtual **cms::Message** * **receive** () throw (cms::CMSException)
Synchronously Receive a Message.
- virtual **cms::Message** * **receive** (int millisecs) throw (cms::CMSException)
Synchronously Receive a Message, time out after defined interval.
- virtual **cms::Message** * **receiveNoWait** () throw (cms::CMSException)
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (**cms::MessageListener** *listener) throw (cms::CMSException)
Sets the MessageListener that this class will send notifys on.
- virtual **cms::MessageListener** * **getMessageListener** () const throw (cms::CMSException)
Gets the MessageListener that this class will send events to.
- virtual std::string **getMessageSelector** () const throw (cms::CMSException)
Gets this message consumer's message selector expression.
- virtual void **acknowledge** (const **Pointer**< **commands::MessageDispatch** > &dispatch) throw (cms::CMSException)
Method called to acknowledge the message passed, called from a message when the mode is client ack.

- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Called asynchronously by the session to dispatch a message.
- void **acknowledge** () throw (cms::CMSEException)
Method called to acknowledge all messages that have been received so far.
- void **commit** () throw (exceptions::ActiveMQException)
Called to Commit the current set of messages in this Transaction.
- void **rollback** () throw (exceptions::ActiveMQException)
Called to Roll back the current set of messages in this Transaction.
- void **doClose** () throw (exceptions::ActiveMQException)
Performs the actual close operation on this consumer.
- const **commands::ConsumerInfo** & **getConsumerInfo** () const
Get the Consumer information for this consumer.
- const **commands::ConsumerId** & **getConsumerId** () const
Get the Consumer Id for this consumer.
- bool **isClosed** () const
- bool **isSynchronizationRegistered** () const
*Has this Consumer Transaction **Synchronization** (p. 3137) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)
*Sets the **Synchronization** (p. 3137) Registered state of this consumer.*
- bool **iterate** ()
Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.
- void **deliverAcks** () throw (exceptions::ActiveMQException)
Forces this consumer to send all pending acks to the broker.
- void **clearMessagesInProgress** ()
Called on a Failover to clear any pending messages.
- long long **getLastDeliveredSequenceId** () const
Gets the currently set Last Delivered Sequence Id.
- void **setLastDeliveredSequenceId** (long long value)
Sets the value of the Last Delivered Sequence Id.

Protected Member Functions

- **Pointer< MessageDispatch > dequeue** (long long timeout) throw (cms::CMSException)

Used by synchronous receive methods to wait for messages to come in.

- **void beforeMessageIsConsumed** (const Pointer< commands::MessageDispatch > &dispatch)

Pre-consume processing.

- **void afterMessageIsConsumed** (const Pointer< commands::MessageDispatch > &dispatch, bool messageExpired)

Post-consume processing.

6.26.1 Constructor & Destructor Documentation

- 6.26.1.1** **activemq::core::ActiveMQConsumer::ActiveMQConsumer** (const Pointer< commands::ConsumerInfo > & *consumerInfo*, ActiveMQSession * *session*, const Pointer< ActiveMQTransactionContext > & *transaction*)

Constructor.

- 6.26.1.2** **virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer** ()
[virtual]

6.26.2 Member Function Documentation

- 6.26.2.1** **void activemq::core::ActiveMQConsumer::acknowledge** () throw (cms::CMSException)

Method called to acknowledge all messages that have been received so far.

Exceptions:

CMSException

- 6.26.2.2** **virtual void activemq::core::ActiveMQConsumer::acknowledge** (const Pointer< commands::MessageDispatch > & *dispatch*) throw (cms::CMSException) [virtual]

Method called to acknowledge the message passed, called from a message when the mode is client ack.

Parameters:

message the Message to Acknowledge

Exceptions:

CMSException

6.26.2.3 `void activemq::core::ActiveMQConsumer::afterMessageIsConsumed (const Pointer< commands::MessageDispatch > & dispatch, bool messageExpired)` [protected]

Post-consume processing.

Parameters:

dispatch - the consumed message

messageExpired - flag indicating if the message has expired.

6.26.2.4 `void activemq::core::ActiveMQConsumer::beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > & dispatch)` [protected]

Pre-consume processing.

Parameters:

dispatch - the message being consumed.

6.26.2.5 `void activemq::core::ActiveMQConsumer::clearMessagesInProgress ()`

Called on a Failover to clear any pending messages.

6.26.2.6 `virtual void activemq::core::ActiveMQConsumer::close () throw (cms::CMSException)` [virtual]

Closes the Consumer. This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions:

CMSException

Implements `cms::Closeable` (p.1021).

6.26.2.7 `void activemq::core::ActiveMQConsumer::commit () throw (exceptions::ActiveMQException)`

Called to Commit the current set of messages in this Transaction.

Exceptions:

ActiveMQException

6.26.2.8 `void activemq::core::ActiveMQConsumer::deliverAcks () throw (exceptions::ActiveMQException)`

Forces this consumer to send all pending acks to the broker.

6.26.2.9 `Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long timeout) throw (cms::CMSException)` [protected]

Used by synchronous receive methods to wait for messages to come in.

Parameters:

timeout - The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.

Returns:

the message, if received within the allotted time. Otherwise NULL.

Exceptions:

InvalidStateException if this consumer is closed upon entering this method.

6.26.2.10 `virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & message)` [virtual]

Called asynchronously by the session to dispatch a message.

Parameters:

message dispatch object pointer

Implements `activemq::core::Dispatcher` (p. 1536).

6.26.2.11 `void activemq::core::ActiveMQConsumer::doClose () throw (exceptions::ActiveMQException)`

Performs the actual close operation on this consumer.

Exceptions:

ActiveMQException

6.26.2.12 `const commands::ConsumerId& activemq::core::ActiveMQConsumer::getConsumerId () const` [inline]

Get the Consumer Id for this consumer.

Returns:

Reference to a Consumer Id Object

6.26.2.13 `const commands::ConsumerInfo& activemq::core::ActiveMQConsumer::getConsumerInfo () const [inline]`

Get the Consumer information for this consumer.

Returns:

Reference to a Consumer Info Object

6.26.2.14 `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId () const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns:

long long containing the sequence id of the last delivered Message.

6.26.2.15 `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener () const throw (cms::CMSException) [inline, virtual]`

Gets the MessageListener that this class will send events to.

Returns:

the currently registered MessageListener interface pointer.

Implements `cms::MessageConsumer` (p. 2215).

6.26.2.16 `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector () const throw (cms::CMSException) [virtual]`

Gets this message consumer's message selector expression.

Returns:

This Consumer's selector expression or "".

Exceptions:

cms::CMSException (p. 1031)

Implements `cms::MessageConsumer` (p. 2215).

6.26.2.17 `bool activemq::core::ActiveMQConsumer::isClosed () const [inline]`

Returns:

if this Consumer has been closed.

6.26.2.18 `bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered ()
const [inline]`

Has this Consumer Transaction **Synchronization** (p. 3137) been added to the transaction.

Returns:

true if the synchronization has been added.

6.26.2.19 `bool activemq::core::ActiveMQConsumer::iterate ()`

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

6.26.2.20 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int
milliseconds) throw (cms::CMSException) [virtual]`

Synchronously Receive a Message, time out after defined interval. Returns null if nothing read.

Parameters:

milliseconds the time in milliseconds to wait before returning

Returns:

new message or null on timeout

Exceptions:

CMSException

Implements `cms::MessageConsumer` (p. 2215).

6.26.2.21 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive ()
throw (cms::CMSException) [virtual]`

Synchronously Receive a Message.

Returns:

new message

Exceptions:

CMSException

Implements `cms::MessageConsumer` (p. 2216).

6.26.2.22 `virtual cms::Message* ac-
tivemq::core::ActiveMQConsumer::receiveNoWait ()
throw (cms::CMSException) [virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns:

new message

Exceptions:

CMSEException

Implements **cms::MessageConsumer** (p. 2216).

6.26.2.23 `void activemq::core::ActiveMQConsumer::rollback () throw (exceptions::ActiveMQException)`

Called to Roll back the current set of messages in this Transaction.

Exceptions:

ActiveMQException

6.26.2.24 `void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters:

value The new value to assign to the Last Delivered Sequence Id property.

6.26.2.25 `virtual void activemq::core::ActiveMQConsumer::setMessageListener (cms::MessageListener * listener) throw (cms::CMSEException) [virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters:

listener MessageListener interface pointer

Implements **cms::MessageConsumer** (p. 2216).

6.26.2.26 `void activemq::core::ActiveMQConsumer::setSynchronizationRegistered (bool value) [inline]`

Sets the **Synchronization** (p. 3137) Registered state of this consumer.

Parameters:

value - true if registered false otherwise.

6.26.2.27 `virtual void activemq::core::ActiveMQConsumer::start () [virtual]`

Starts the Consumer if not already started and not closed. A consumer will no deliver messages until started.

6.26.2.28 virtual void activemq::core::ActiveMQConsumer::stop () [virtual]

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again. A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConsumer.h`

6.27 activemq::library::ActiveMQCPP Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

Public Member Functions

- virtual `~ActiveMQCPP ()`

Static Public Member Functions

- static void `initializeLibrary ()`
Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.
- static void `initializeLibrary (int argc, char **argv)`
Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.
- static void `shutdownLibrary ()`
Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

Protected Member Functions

- `ActiveMQCPP ()`
- `ActiveMQCPP (const ActiveMQCPP &)`
- `ActiveMQCPP & operator= (const ActiveMQCPP &)`

6.27.1 Constructor & Destructor Documentation

6.27.1.1 `activemq::library::ActiveMQCPP::ActiveMQCPP () [inline, protected]`

6.27.1.2 `activemq::library::ActiveMQCPP::ActiveMQCPP (const ActiveMQCPP &) [protected]`

6.27.1.3 `virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP () [inline, virtual]`

6.27.2 Member Function Documentation

6.27.2.1 `static void activemq::library::ActiveMQCPP::initializeLibrary (int argc, char ** argv) [static]`

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library. This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

Parameters:

argc - the count of arguments passed to this Process.

argv - the array of string arguments passed to this process.

Exceptions:

runtime_error if an error occurs while initializing this library.

6.27.2.2 static void activemq::library::ActiveMQCPP::initializeLibrary () [static]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

Exceptions:

runtime_error if an error occurs while initializing this library.

6.27.2.3 ActiveMQCPP& activemq::library::ActiveMQCPP::operator= (const ActiveMQCPP &) [protected]

6.27.2.4 static void activemq::library::ActiveMQCPP::shutdownLibrary () [static]

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point. All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- src/main/activemq/library/**ActiveMQCPP.h**

6.28 activemq::commands::ActiveMQDestination Class Reference

#include <src/main/activemq/commands/ActiveMQDestination.h> Inheritance diagram for activemq::commands::ActiveMQDestination:

Data Structures

- struct **DestinationFilter**

Public Member Functions

- **ActiveMQDestination** ()
- **ActiveMQDestination** (const std::string &**physicalName**)
- virtual ~**ActiveMQDestination** ()
- virtual **ActiveMQDestination** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual const std::string & **getPhysicalName** () const
Fetch this destination's physical name.
- virtual std::string & **getPhysicalName** ()
- virtual void **setPhysicalName** (const std::string &**physicalName**)
Set this destination's physical name.
- virtual bool **isAdvisory** () const
- virtual void **setAdvisory** (bool **advisory**)
- virtual bool **isConsumerAdvisory** () const
- virtual bool **isProducerAdvisory** () const
- virtual bool **isConnectionAdvisory** () const
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isOrdered** () const

- virtual void **setOrdered** (bool **ordered**)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &**orderedTarget**)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0
Returns the Type of Destination that this object represents.
- virtual bool **isTemporary** () const
Returns true if a temporary Destination.
- virtual bool **isTopic** () const
Returns true if a Topic Destination.
- virtual bool **isQueue** () const
Returns true if a Queue Destination.
- virtual bool **isComposite** () const
Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.
- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** * **getCMSDestination** () const

Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)
Create a temporary name from the clientId.
- static std::string **getClientId** (const **ActiveMQDestination** *destination)
From a temporary destination find the clientId of the Connection that created it.
- static **Pointer< ActiveMQDestination >** **createDestination** (int type, const std::string &name)
Creates a Destination given the String Name to use and a Type.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQDESTINATION** = 0

Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

Static Protected Attributes

- static const std::string **ADVISORY_PREFIX**
prefix for Advisory message destinations
- static const std::string **CONSUMER_ADVISORY_PREFIX**
prefix for consumer advisory destinations
- static const std::string **PRODUCER_ADVISORY_PREFIX**
prefix for producer advisory destinations
- static const std::string **CONNECTION_ADVISORY_PREFIX**
prefix for connection advisory destinations
- static const std::string **DEFAULT_ORDERED_TARGET**
The default target for ordered destinations.
- static const std::string **TEMP_PREFIX**
- static const std::string **TEMP_POSTFIX**
- static const std::string **COMPOSITE_SEPARATOR**
- static const std::string **QUEUE_QUALIFIED_PREFIX**
- static const std::string **TOPIC_QUALIFIED_PREFIX**
- static const std::string **TEMP_QUEUE_QUALIFIED_PREFIX**
- static const std::string **TEMP_TOPIC_QUALIFIED_PREFIX**

6.28.1 Constructor & Destructor Documentation

6.28.1.1 `activemq::commands::ActiveMQDestination::ActiveMQDestination ()`

6.28.1.2 `activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & physicalName)`

6.28.1.3 `virtual
activemq::commands::ActiveMQDestination::~~ActiveMQDestination ()
[inline, virtual]`

6.28.2 Member Function Documentation

6.28.2.1 `virtual ActiveMQDestination* ac-
tivemq::commands::ActiveMQDestination::cloneDataStructure () const
[inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 420),
`activemq::commands::ActiveMQTempDestination` (p. 500), **ac-**
`tivemq::commands::ActiveMQTempQueue` (p. 524), **ac-**
`tivemq::commands::ActiveMQTempTopic` (p. 549), and **ac-**
`tivemq::commands::ActiveMQTopic` (p. 599).

6.28.2.2 virtual void `activemq::commands::ActiveMQDestination::copyDataStructure` (const `DataStructure * src`) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

`src` - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 420),
`activemq::commands::ActiveMQTempDestination` (p. 500), **ac-**
`tivemq::commands::ActiveMQTempQueue` (p. 525), **ac-**
`tivemq::commands::ActiveMQTempTopic` (p. 550), and **ac-**
`tivemq::commands::ActiveMQTopic` (p. 599).

Referenced by `activemq::commands::ActiveMQTempDestination::copyDataStructure()`.

6.28.2.3 static `Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination` (int *type*, const std::string & *name*) [static]

Creates a Destination given the String Name to use and a Type.

Parameters:

type - The Type of Destination to Create

name - The Name to use in the creation of the Destination

Returns:

Pointer to a new `ActiveMQDestination` (p. 274) instance.

6.28.2.4 static std::string `activemq::commands::ActiveMQDestination::createTemporaryName` (const std::string & *clientId*) [inline, static]

Create a temporary name from the clientId.

Parameters:

clientId

Returns:

6.28.2.5 `virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 421),
activemq::commands::ActiveMQTempDestination (p. 501), **ac-**
tivemq::commands::ActiveMQTempQueue (p. 525), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 550), and **ac-**
tivemq::commands::ActiveMQTopic (p. 600).

Referenced by **activemq::commands::ActiveMQTopic::equals()**, and **ac-**
tivemq::commands::ActiveMQTempDestination::equals().

6.28.2.6 `static std::string activemq::commands::ActiveMQDestination::getClientId (const ActiveMQDestination * destination)` [static]

From a temporary destination find the clientId of the Connection that created it.

Parameters:

destination

Returns:

the clientId or null if not a temporary destination

6.28.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination () const` [inline, virtual]

Returns:

the **cms::Destination** (p. 1480) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 421),
activemq::commands::ActiveMQTempQueue (p. 525), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 550), and **ac-**
tivemq::commands::ActiveMQTopic (p. 600).

6.28.2.8 `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType () const` [virtual]

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1464).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 421),
activemq::commands::ActiveMQTempDestination (p. 501), **ac-**
tivemq::commands::ActiveMQTempQueue (p. 526), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 551), and **ac-**
tivemq::commands::ActiveMQTopic (p. 600).

6.28.2.9 `virtual cms::Destination::DestinationType ac-`
`tivemq::commands::ActiveMQDestination::getDestinationType () const`
`[pure virtual]`

Returns the Type of Destination that this object represents.

Returns:

int type qualifier.

Implemented in **activemq::commands::ActiveMQQueue** (p. 422),
activemq::commands::ActiveMQTempQueue (p. 526), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 551), and **ac-**
tivemq::commands::ActiveMQTopic (p. 601).

6.28.2.10 `const activemq::util::ActiveMQProperties& ac-`
`tivemq::commands::ActiveMQDestination::getOptions () const`
`[inline]`

Returns:

a reference (const) to the options properties for this Destination.

6.28.2.11 `virtual std::string ac-`
`tivemq::commands::ActiveMQDestination::getOrderedTarget () const`
`[inline, virtual]`

Returns:

Returns the orderedTarget.

6.28.2.12 `virtual std::string& ac-`
`tivemq::commands::ActiveMQDestination::getPhysicalName () [inline,`
`virtual]`

6.28.2.13 `virtual const std::string& ac-`
`tivemq::commands::ActiveMQDestination::getPhysicalName () const`
`[inline, virtual]`

Fetch this destination's physical name.

Returns:

const string containing the name

6.28.2.14 `virtual bool activemq::commands::ActiveMQDestination::isAdvisory () const [inline, virtual]`

Returns:

Returns the advisory.

6.28.2.15 `virtual bool activemq::commands::ActiveMQDestination::isComposite () const [inline, virtual]`

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

Returns:

true if this destination represents a collection of child destinations.

6.28.2.16 `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory () const [inline, virtual]`

Returns:

true if this is a destination for Connection advisories

6.28.2.17 `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory () const [inline, virtual]`

Returns:

true if this is a destination for Consumer advisories

6.28.2.18 `virtual bool activemq::commands::ActiveMQDestination::isExclusive () const [inline, virtual]`

Returns:

Returns the exclusive.

6.28.2.19 `virtual bool activemq::commands::ActiveMQDestination::isOrdered () const [inline, virtual]`

Returns:

Returns the ordered.

6.28.2.20 `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory () const`
[inline, virtual]

Returns:

true if this is a destination for Producer advisories

6.28.2.21 `virtual bool activemq::commands::ActiveMQDestination::isQueue () const`
[inline, virtual]

Returns true if a Queue Destination.

Returns:

true/false

6.28.2.22 `virtual bool activemq::commands::ActiveMQDestination::isTemporary () const`
[inline, virtual]

Returns true if a temporary Destination.

Returns:

true/false

References cms::Destination::TEMPORARY_QUEUE, and cms::Destination::TEMPORARY_TOPIC.

6.28.2.23 `virtual bool activemq::commands::ActiveMQDestination::isTopic () const`
[inline, virtual]

Returns true if a Topic Destination.

Returns:

true/false

References cms::Destination::TEMPORARY_TOPIC, and cms::Destination::TOPIC.

6.28.2.24 `virtual bool activemq::commands::ActiveMQDestination::isWildcard () const`
[inline, virtual]

Returns:

true if the destination matches multiple possible destinations

6.28.2.25 `virtual void activemq::commands::ActiveMQDestination::setAdvisory (bool advisory)` [inline, virtual]

Parameters:

advisory The advisory to set.

6.28.2.26 `virtual void activemq::commands::ActiveMQDestination::setExclusive`
`(bool exclusive)` [inline, virtual]

Parameters:

exclusive The exclusive to set.

6.28.2.27 `virtual void activemq::commands::ActiveMQDestination::setOrdered`
`(bool ordered)` [inline, virtual]

Parameters:

ordered The ordered to set.

6.28.2.28 `virtual void ac-`
`tivemq::commands::ActiveMQDestination::setOrderedTarget (const`
`std::string & orderedTarget)` [inline, virtual]

Parameters:

orderedTarget The orderedTarget to set.

6.28.2.29 `virtual void ac-`
`tivemq::commands::ActiveMQDestination::setPhysicalName (const`
`std::string & physicalName)` [virtual]

Set this destination's physical name.

Returns:

const string containing the name

6.28.2.30 `virtual std::string activemq::commands::ActiveMQDestination::toString`
`() const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 718).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 422),
`activemq::commands::ActiveMQTempDestination` (p. 501), `ac-`
`tivemq::commands::ActiveMQTempQueue` (p. 526), `ac-`
`tivemq::commands::ActiveMQTempTopic` (p. 551), and `ac-`
`tivemq::commands::ActiveMQTopic` (p. 601).

6.28.3 Field Documentation

6.28.3.1 `bool activemq::commands::ActiveMQDestination::advisory` [protected]

6.28.3.2 `const std::string
activemq::commands::ActiveMQDestination::ADVISORY_
PREFIX` [static, protected]

prefix for Advisory message destinations

6.28.3.3 `const std::string
activemq::commands::ActiveMQDestination::COMPOSITE_
SEPARATOR` [static, protected]

6.28.3.4 `const std::string
activemq::commands::ActiveMQDestination::CONNECTION_
ADVISORY_PREFIX` [static, protected]

prefix for connection advisory destinations

6.28.3.5 `const std::string
activemq::commands::ActiveMQDestination::CONSUMER_
ADVISORY_PREFIX` [static, protected]

prefix for consumer advisory destinations

6.28.3.6 `const std::string
activemq::commands::ActiveMQDestination::DEFAULT_
ORDERED_TARGET` [static, protected]

The default target for ordered destinations.

6.28.3.7 `bool activemq::commands::ActiveMQDestination::exclusive` [protected]

6.28.3.8 `const unsigned char activemq::commands::ActiveMQDestination::ID _ - ACTIVEMQDESTINATION = 0` [static]

6.28.3.9 `util::ActiveMQProperties activemq::commands::ActiveMQDestination::options` [protected]

6.28.3.10 `bool activemq::commands::ActiveMQDestination::ordered` [protected]

6.28.3.11 `std::string activemq::commands::ActiveMQDestination::orderedTarget` [protected]

6.28.3.12 `std::string activemq::commands::ActiveMQDestination::physicalName` [protected]

6.28.3.13 `const std::string activemq::commands::ActiveMQDestination::PRODUCER _ - ADVISORY _ PREFIX` [static, protected]

prefix for producer advisory destinations

6.28.3.14 `const std::string activemq::commands::ActiveMQDestination::QUEUE _ - QUALIFIED _ PREFIX` [static, protected]

6.28.3.15 `const std::string activemq::commands::ActiveMQDestination::TEMP _ - POSTFIX` [static, protected]

6.28.3.16 `const std::string activemq::commands::ActiveMQDestination::TEMP _ - PREFIX` [static, protected]

6.28.3.17 `const std::string activemq::commands::ActiveMQDestination::TEMP _ - QUEUE _ QUALIFIED _ PREFIX` [static, protected]

6.28.3.18 `const std::string activemq::commands::ActiveMQDestination::TEMP _ - TOPIC _ QUALIFIED _ PREFIX` [static, protected]

6.28.3.19 `const std::string activemq::commands::ActiveMQDestination::TOPIC _ - QUALIFIED _ PREFIX` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

6.29 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 285).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller:
```

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.29.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 285).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.29.2 Constructor & Destructor Documentation

6.29.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

6.29.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

6.29.3 Member Function Documentation

6.29.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 424), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 504), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 529), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 554), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 603).

6.29.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 425), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 504), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 530), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 555), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 604).

6.29.3.3 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 425), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 505), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 530), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 555), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 604).

6.29.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 425), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 505), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 530), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 555), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 604).

6.29.3.5 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 426), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 531), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 556), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 605).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h`

6.30 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 289).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.30.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 289).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.30.2 Constructor & Destructor Documentation

6.30.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

6.30.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

6.30.3 Member Function Documentation

6.30.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 428), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 508), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 558), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 607).

6.30.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 429), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 508), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 534), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 608).

6.30.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 429), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 534), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 608).

6.30.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 429), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 534), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 608).

6.30.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 430), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 510), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 535), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 560), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h`

6.31 activemq:wireformat::openwire::marshal:v4::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h> Inheritance diagram for activemq:wireformat::openwire::marshal:v4::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.31.2 Constructor & Destructor Documentation

6.31.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

6.31.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

6.31.3 Member Function Documentation

6.31.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 432), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 512), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 562), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 611).

6.31.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 433), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 512), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 538), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 612).

6.31.3.3 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 433), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 513), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 538), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 612).

6.31.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 433), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 513), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 538), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 612).

6.31.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat*** - describes the wire format of the broker.
- dataStructure*** - Object to be un-marshaled.
- dataIn*** - BinaryReader that provides that data.
- bs*** - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 434), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 514), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 539), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 564), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 613).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h`

6.32 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.32.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

6.32.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

6.32.3 Member Function Documentation

6.32.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 436), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 516), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 541), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 566), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 615).

6.32.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 437), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 516), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 542), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 567), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 616).

6.32.3.3 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 437), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 517), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 542), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 567), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 616).

6.32.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 437), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 517), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 542), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 567), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 616).

6.32.3.5 virtual void `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 438), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 518), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 543), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 568), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 617).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h`

6.33 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.33.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.33.2 Constructor & Destructor Documentation

6.33.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

6.33.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

6.33.3 Member Function Documentation

6.33.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 440), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 520), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 545), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 570), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 619).

6.33.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 441), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 520), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 546), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 571), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 620).

6.33.3.3 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 441), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 521), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 546), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 571), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 620).

6.33.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 441), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 521), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 546), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 571), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 620).

6.33.3.5 virtual void `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 442), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 522), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 547), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 572), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 621).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h`

6.34 activemq::exceptions::ActiveMQException Class Reference

#include <src/main/activemq/exceptions/ActiveMQException.h> Inheritance diagram for activemq::exceptions::ActiveMQException:

Public Member Functions

- **ActiveMQException** () throw ()
Default Constructor.
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** * **clone** () const
Clones this exception.
- virtual **cms::CMSException** **convertToCMSException** () const
Converts this exception to a new CMSException.

6.34.1 Constructor & Destructor Documentation

6.34.1.1 activemq::exceptions::ActiveMQException::ActiveMQException () throw ()

Default Constructor.

6.34.1.2 activemq::exceptions::ActiveMQException::ActiveMQException (const ActiveMQException & ex) throw ()

Copy Constructor.

Parameters:

ex The Exception whose internal data is copied into this instance.

6.34.1.3 `activemq::exceptions::ActiveMQException::ActiveMQException (const decaf::lang::Exception & ex) throw ()`

Copy Constructor.

Parameters:

ex The Exception whose internal data is copied into this instance.

6.34.1.4 `activemq::exceptions::ActiveMQException::ActiveMQException (const char * file, const int lineNumber, const char * msg, ...) throw ()`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message.

Parameters:

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

msg The message to report.

... The list of primitives that are formatted into the message.

6.34.1.5 `virtual activemq::exceptions::ActiveMQException::~~ActiveMQException () throw () [virtual]`

6.34.2 Member Function Documentation

6.34.2.1 `virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone () const [virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

Copy of this Exception object

Reimplemented from `decaf::lang::Exception` (p. 1577).

Reimplemented in `activemq::exceptions::BrokerException` (p. 749).

6.34.2.2 `virtual cms::CMSEException activemq::exceptions::ActiveMQException::convertToCMSEException () const [virtual]`

Converts this exception to a new CMSEException.

Returns:

a CMSEException with the data from this exception

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/ActiveMQException.h`

6.35 activemq::commands::ActiveMQMapMessage Class Reference

#include <src/main/activemq/commands/ActiveMQMapMessage.h> Inheritance diagram for activemq::commands::ActiveMQMapMessage:

Public Member Functions

- **ActiveMQMapMessage** ()
- virtual **~ActiveMQMapMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual bool **isMarshalAware** () const
Determine if this object is aware of marshaling and should have its before and after marshaling methods called.
- virtual **ActiveMQMapMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat) throw (decaf::io::IOException)
Perform any processing needed before an marshal.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSEException)
Clears out the body of the message.
- virtual **cms::MapMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual std::vector< std::string > **getMapNames** () const throw (cms::CMSEException)
Returns an Enumeration of all the names in the MapMessage object.
- virtual bool **itemExists** (const std::string &name) const throw (cms::CMSEException)
Indicates whether an item exists in this MapMessage object.

- virtual bool **getBoolean** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (const std::string &name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a boolean value with the specified name into the Map.
- virtual unsigned char **getByte** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Byte value of the Specified name.
- virtual void **setByte** (const std::string &name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Byte value with the specified name into the Map.
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Bytes value of the Specified name.
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Bytes value with the specified name into the Map.
- virtual char **getChar** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Char value of the Specified name.
- virtual void **setChar** (const std::string &name, char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Double value of the Specified name.
- virtual void **setDouble** (const std::string &name, double value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Double value with the specified name into the Map.
- virtual float **getFloat** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Float value of the Specified name.
- virtual void **setFloat** (const std::string &name, float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Float value with the specified name into the Map.
- virtual int **getInt** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Int value of the Specified name.

- virtual void **setInt** (const std::string &name, int value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a String value with the specified name into the Map.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMAPMESSAGE** = 25

Protected Member Functions

- **util::PrimitiveMap & getMap** () throw (decaf::lang::exceptions::NullPointerException)
Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.
- const **util::PrimitiveMap & getMap** () const throw (decaf::lang::exceptions::NullPointerException)
- virtual void **checkMapIsUnmarshalled** () const throw (decaf::lang::exceptions::NullPointerException)
Performs the unmarshal on the Map if needed, otherwise just returns.

6.35.1 Constructor & Destructor Documentation

6.35.1.1 `activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ()`

6.35.1.2 `virtual
activemq::commands::ActiveMQMapMessage::~~ActiveMQMapMessage ()
[virtual]`

6.35.2 Member Function Documentation

6.35.2.1 `virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal
(wireformat::WireFormat * wireFormat) throw (decaf::io::IOException)
[virtual]`

Perform any processing needed before an marshal.

Parameters:

wireFormat - the OpenWireFormat object in use.

Implements `activemq::wireformat::MarshalAware` (p. 2119).

6.35.2.2 `virtual void ac-
tivemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled ()
const throw (decaf::lang::exceptions::NullPointerException) [protected,
virtual]`

Performs the unmarshal on the Map if needed, otherwise just returns.

6.35.2.3 `virtual void activemq::commands::ActiveMQMapMessage::clearBody ()
throw (cms::CMSException) [virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 368).

6.35.2.4 `virtual cms::MapMessage* ac-
tivemq::commands::ActiveMQMapMessage::clone () const
[inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements `cms::Message` (p. 2168).

6.35.2.5 `virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2149).

6.35.2.6 `virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2150).

6.35.2.7 `virtual bool activemq::commands::ActiveMQMapMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 369).

6.35.2.8 `virtual bool activemq::commands::ActiveMQMapMessage::getBoolean (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Boolean value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2108).

6.35.2.9 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Byte value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2109).

6.35.2.10 `virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Bytes value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2109).

6.35.2.11 `virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Char value of the Specified name.

Parameters:

name name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2109).

6.35.2.12 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from **activemq::commands::Message** (p. 2152).

6.35.2.13 `virtual double activemq::commands::ActiveMQMapMessage::getDouble (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Double value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2110).

6.35.2.14 `virtual float activemq::commands::ActiveMQMapMessage::getFloat (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Float value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2110).

6.35.2.15 `virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Int value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2110).

6.35.2.16 `virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & name) const throw (cms::MessageFormatException, cms::CMSEException)` [virtual]

Returns the Long value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2111).

6.35.2.17 `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () const throw (decaf::lang::exceptions::NullPointerException)` [protected]

6.35.2.18 `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () throw (decaf::lang::exceptions::NullPointerException)` [protected]

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

Returns:

reference to a PrimitiveMap.

6.35.2.19 `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames () const throw (cms::CMSEException)` [virtual]

Returns an Enumeration of all the names in the MapMessage object.

Returns:

STL Vector of String values, each of which is the name of an item in the MapMessage

Exceptions:

CMSEException - if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 2111).

6.35.2.20 `virtual short activemq::commands::ActiveMQMapMessage::getShort
(const std::string & name) const throw (cms::MessageFormatException,
cms::CMSException) [virtual]`

Returns the Short value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2111).

6.35.2.21 `virtual std::string ac-
tivismq::commands::ActiveMQMapMessage::getString
(const std::string & name) const throw (cms::MessageFormatException,
cms::CMSException) [virtual]`

Returns the String value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implements `cms::MapMessage` (p. 2111).

6.35.2.22 `virtual bool ac-
tivismq::commands::ActiveMQMapMessage::isMarshalAware () const
[inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

Returns:

true if aware of marshaling

Reimplemented from `activemq::commands::Message` (p. 2155).

6.35.2.23 `virtual bool activemq::commands::ActiveMQMapMessage::itemExists
(const std::string & name) const throw (cms::CMSException)
[virtual]`

Indicates whether an item exists in this MapMessage object.

Parameters:

name String name of the Object in question

Returns:

boolean value indicating if the name is in the map

Exceptions:

CMSEException - if the operation fails due to an internal error.

Implements **cms::MapMessage** (p. 2112).

6.35.2.24 `virtual void activemq::commands::ActiveMQMapMessage::setBoolean (const std::string & name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)` [virtual]

Sets a boolean value with the specified name into the Map.

Parameters:

name the name of the boolean

value the boolean value to set in the Map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the **Message** (p. 2145) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2112).

6.35.2.25 `virtual void activemq::commands::ActiveMQMapMessage::setByte (const std::string & name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)` [virtual]

Sets a Byte value with the specified name into the Map.

Parameters:

name the name of the Byte

value the Byte value to set in the Map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the **Message** (p. 2145) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2112).

6.35.2.26 `virtual void activemq::commands::ActiveMQMapMessage::setBytes (const std::string & name, const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets a Bytes value with the specified name into the Map.

Parameters:

name The name of the Bytes

value The Bytes value to set in the Map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2145) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2113).

6.35.2.27 `virtual void activemq::commands::ActiveMQMapMessage::setChar (const std::string & name, char value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets a Char value with the specified name into the Map.

Parameters:

name the name of the Char

value the Char value to set in the Map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2145) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2113).

6.35.2.28 `virtual void activemq::commands::ActiveMQMapMessage::setDouble (const std::string & name, double value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets a Double value with the specified name into the Map.

Parameters:

name The name of the Double

value The Double value to set in the Map

Exceptions:

CMSException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2145) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2114).

6.35.2.29 `virtual void activemq::commands::ActiveMQMapMessage::setFloat (const std::string & name, float value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Float value with the specified name into the Map.

Parameters:

name The name of the Float

value The Float value to set in the Map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2145) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2114).

6.35.2.30 `virtual void activemq::commands::ActiveMQMapMessage::setInt (const std::string & name, int value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Int value with the specified name into the Map.

Parameters:

name The name of the Int

value The Int value to set in the Map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2145) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2114).

6.35.2.31 `virtual void activemq::commands::ActiveMQMapMessage::setLong (const std::string & name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Long value with the specified name into the Map.

Parameters:

name The name of the Long

value The Long value to set in the Map

Exceptions:

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2145) is in Read-only Mode.

Implements `cms::MapMessage` (p. 2115).

6.35.2.32 `virtual void activemq::commands::ActiveMQMapMessage::setShort (const std::string & name, short value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets a Short value with the specified name into the Map.

Parameters:

name The name of the Short
value The Short value to set in the Map

Exceptions:

CMSException - if the operation fails due to an internal error.
MessageNotWriteableException - if the **Message** (p. 2145) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2115).

6.35.2.33 `virtual void activemq::commands::ActiveMQMapMessage::setString (const std::string & name, const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets a String value with the specified name into the Map.

Parameters:

name The name of the String
value The String value to set in the Map

Exceptions:

CMSException - if the operation fails due to an internal error.
MessageNotWriteableException - if the **Message** (p. 2145) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2115).

6.35.2.34 `virtual std::string activemq::commands::ActiveMQMapMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2159).

6.35.3 Field Documentation

6.35.3.1 `const unsigned char activemq::commands::ActiveMQMapMessage::ID_ - ACTIVEMQMAPMESSAGE = 25 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMapMessage.h`

6.36 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 322).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.36.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 322).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.36.2 Constructor & Destructor Documentation

6.36.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` `[inline]`

6.36.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` `[inline, virtual]`

6.36.3 Member Function Documentation

6.36.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject(const std::string & name) const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.36.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.36.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.36.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.36.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2317).

6.36.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

6.36.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h`

6.37 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 326).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.37.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 326).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.37.2 Constructor & Destructor Documentation

6.37.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` `[inline]`

6.37.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` `[inline, virtual]`

6.37.3 Member Function Documentation

6.37.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.37.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.37.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.37.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.37.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2327).

6.37.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2328).

6.37.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2328).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h

6.38 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 330).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.38.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 330).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.38.2 Constructor & Destructor Documentation

6.38.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` `[inline]`

6.38.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` `[inline, virtual]`

6.38.3 Member Function Documentation

6.38.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.38.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.38.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.38.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.38.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2312).

6.38.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

6.38.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h`

6.39 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 334).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.39.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 334).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.39.2 Constructor & Destructor Documentation

6.39.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` `[inline]`

6.39.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` `[inline, virtual]`

6.39.3 Member Function Documentation

6.39.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.39.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.39.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.39.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.39.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2322).

6.39.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

6.39.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h`

6.40 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 338).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 338).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.40.2 Constructor & Destructor Documentation

6.40.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

6.40.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

6.40.3 Member Function Documentation

6.40.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.40.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.40.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.40.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.40.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2307).

6.40.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

6.40.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h`

6.41 activemq::commands::ActiveMQMessage Class Reference

#include <src/main/activemq/commands/ActiveMQMessage.h> Inheritance diagram for activemq::commands::ActiveMQMessage:

Public Member Functions

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **ActiveMQMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMESSAGE** = 23

6.41.1 Constructor & Destructor Documentation

6.41.1.1 **activemq::commands::ActiveMQMessage::ActiveMQMessage** ()

6.41.1.2 **virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage** ()
[inline, virtual]

6.41.2 Member Function Documentation

6.41.2.1 **virtual cms::Message* activemq::commands::ActiveMQMessage::clone** ()
const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 2168).

6.41.2.2 `virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2149).

6.41.2.3 `virtual void activemq::commands::ActiveMQMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2150).

6.41.2.4 `virtual bool activemq::commands::ActiveMQMessage::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 369).

6.41.2.5 `virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from **activemq::commands::Message** (p. 2152).

6.41.2.6 `virtual std::string activemq::commands::ActiveMQMessage::toString ()` `const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p.2159).

6.41.3 Field Documentation

6.41.3.1 `const unsigned char activemq::commands::ActiveMQMessage::ID_-` `ACTIVEMQMESSAGE = 23` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessage.h`

6.42 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 345).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 345). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

6.42.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

6.42.3 Member Function Documentation

6.42.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.42.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.42.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.42.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.42.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2317).

6.42.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

6.42.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h`

6.43 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 349).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.43.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 349). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.43.2 Constructor & Destructor Documentation

6.43.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

6.43.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

6.43.3 Member Function Documentation

6.43.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.43.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.43.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.43.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.43.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2327).

6.43.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

6.43.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h`

6.44 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 353).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.44.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 353). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.44.2 Constructor & Destructor Documentation

6.44.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

6.44.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

6.44.3 Member Function Documentation

6.44.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.44.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.44.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.44.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.44.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2312).

6.44.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

6.44.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h`

6.45 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 357).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.45.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 357). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.45.2 Constructor & Destructor Documentation

6.45.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

6.45.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

6.45.3 Member Function Documentation

6.45.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.45.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.45.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.45.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.45.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2322).

6.45.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

6.45.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h`

6.46 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 361).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 361). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller()` [inline]

6.46.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller()` [inline, virtual]

6.46.3 Member Function Documentation

6.46.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.46.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.46.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.46.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.46.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2307).

6.46.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

6.46.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h`

6.47 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference

#include <src/main/activemq/commands/ActiveMQMessageTemplate.h> Inheritance diagram for activemq::commands::ActiveMQMessageTemplate< T >:

Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual **~ActiveMQMessageTemplate** ()
- virtual void **acknowledge** () const throw (cms::IllegalStateException, cms::CMSEException)
Acknowledges all consumed messages of the session of this consumed message.
- virtual void **onSend** ()
*Resets the **Message** (p. 2145) to a Read-Only state.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSEException)
Clears out the body of the message.
- virtual void **clearProperties** () throw (cms::CMSEException)
Clears the message properties.
- virtual std::vector< std::string > **getPropertyNames** () const throw (cms::CMSEException)
Retrieves the property names.
- virtual bool **propertyExists** (const std::string &name) const throw (cms::CMSEException)
Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Gets a boolean property.
- virtual unsigned char **getByteProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a long property.

- virtual short **getShortProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a short property.

- virtual std::string **getStringProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a string property.

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a boolean property.

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a byte property.

- virtual void **setDoubleProperty** (const std::string &name, double value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a int property.

- virtual void **setLongProperty** (const std::string &name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a long property.

- virtual void **setShortProperty** (const std::string &name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a short property.

- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a string property.

- virtual std::string **getCMSCorrelationID** () const throw (cms::CMSEException)

Get the Correlation Id for this message.

- virtual void **setCMSCorrelationID** (const std::string &correlationId) throw (cms::CMSException)

Sets the Correlation Id used by this message.

- virtual int **getCMSDeliveryMode** () const throw (cms::CMSException)

Gets the DeliveryMode for this message.

- virtual void **setCMSDeliveryMode** (int mode) throw (cms::CMSException)

Sets the DeliveryMode for this message.

- virtual const cms::Destination * **getCMSDestination** () const throw (cms::CMSException)

*Gets the Destination for this **Message** (p. 2145), returns a.*

- virtual void **setCMSDestination** (const cms::Destination *destination) throw (cms::CMSException)

Sets the Destination for this message.

- virtual long long **getCMSExpiration** () const throw (cms::CMSException)

*Gets the Expiration Time for this **Message** (p. 2145).*

- virtual void **setCMSExpiration** (long long expireTime) throw (cms::CMSException)

Sets the Expiration Time for this message.

- virtual std::string **getCMSMessageID** () const throw (cms::CMSException)

*Gets the CMS **Message** (p. 2145) Id for this **Message** (p. 2145).*

- virtual void **setCMSMessageID** (const std::string &id AMQCPP_UNUSED) throw (cms::CMSException)

*Sets the CMS **Message** (p. 2145) Id for this message.*

- virtual int **getCMSPriority** () const throw (cms::CMSException)

*Gets the Priority Value for this **Message** (p. 2145).*

- virtual void **setCMSPriority** (int priority) throw (cms::CMSException)

Sets the Priority Value for this message.

- virtual bool **getCMSRedelivered** () const throw (cms::CMSException)

*Gets the Redelivered Flag for this **Message** (p. 2145).*

- virtual void **setCMSRedelivered** (bool redelivered AMQCPP_UNUSED) throw (cms::CMSException)

Sets the Redelivered Flag for this message.

- virtual const cms::Destination * **getCMSReplyTo** () const throw (cms::CMSException)

*Gets the CMS Reply To Address for this **Message** (p. 2145).*

- virtual void **setCMSReplyTo** (const **cms::Destination** ***destination**) throw (cms::CMSException)
Sets the CMS Reply To Address for this message.
- virtual long long **getCMSTimestamp** () const throw (cms::CMSException)
*Gets the Time Stamp for this **Message** (p. 2145).*
- virtual void **setCMSTimestamp** (long long timeStamp) throw (cms::CMSException)
Sets the Time Stamp for this message.
- virtual std::string **getCMSType** () const throw (cms::CMSException)
*Gets the CMS **Message** (p. 2145) Type for this **Message** (p. 2145).*
- virtual void **setCMSType** (const std::string &**type**) throw (cms::CMSException)
*Sets the CMS **Message** (p. 2145) Type for this message.*

Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T
>
```

6.47.1 Constructor & Destructor Documentation

- 6.47.1.1 `template<typename T>
activemq::commands::ActiveMQMessageTemplate< T
>::ActiveMQMessageTemplate () [inline]`
- 6.47.1.2 `template<typename T> virtual
activemq::commands::ActiveMQMessageTemplate< T
>::~~ActiveMQMessageTemplate () [inline, virtual]`

6.47.2 Member Function Documentation

- 6.47.2.1 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::acknowledge () const throw (cms::IllegalStateException,
cms::CMSException) [inline, virtual]`

Acknowledges all consumed messages of the session of this consumed message.

- 6.47.2.2 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::clearBody () throw (cms::CMSException) [inline, virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 200), **activemq::commands::ActiveMQMapMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 467), and **activemq::commands::ActiveMQTextMessage** (p. 574).

6.47.2.3 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::clearProperties () throw (cms::CMSException) [inline, virtual]`

Clears the message properties. Does not clear the body or header values.

6.47.2.4 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::equals (const DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 2150).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 174), **activemq::commands::ActiveMQBytesMessage** (p. 201), **activemq::commands::ActiveMQMapMessage** (p. 312), **activemq::commands::ActiveMQMessage** (p. 343), **activemq::commands::ActiveMQObjectMessage** (p. 384), **activemq::commands::ActiveMQStreamMessage** (p. 468), and **activemq::commands::ActiveMQTextMessage** (p. 575).

6.47.2.5 `template<typename T> void
activemq::commands::ActiveMQMessageTemplate< T
>::failIfReadOnlyBody () const [inline, protected]`

6.47.2.6 `template<typename T> void
activemq::commands::ActiveMQMessageTemplate< T
>::failIfReadOnlyProperties () const [inline, protected]`

6.47.2.7 `template<typename T> void
activemq::commands::ActiveMQMessageTemplate< T
>::failIfWriteOnlyBody () const [inline, protected]`

6.47.2.8 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::getBooleanProperty (const std::string & name) const throw (
cms::MessageFormatException, cms::CMSException) [inline, virtual]`

Gets a boolean property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.9  template<typename T> virtual unsigned char
          activemq::commands::ActiveMQMessageTemplate< T >::getByteProperty
          (const std::string & name) const throw ( cms::MessageFormatException,
          cms::CMSEException ) [inline, virtual]
```

Gets a byte property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.10 template<typename T> virtual std::string
          activemq::commands::ActiveMQMessageTemplate< T
          >::getCMSCorrelationID () const throw ( cms::CMSEException )
          [inline, virtual]
```

Get the Correlation Id for this message.

Returns:

string representation of the correlation Id

Exceptions:

CMSEException

```
6.47.2.11 template<typename T> virtual int
          activemq::commands::ActiveMQMessageTemplate< T
          >::getCMSDeliveryMode () const throw ( cms::CMSEException )
          [inline, virtual]
```

Gets the DeliveryMode for this message.

Returns:

DeliveryMode enumerated value.

Exceptions:

CMSException

6.47.2.12 `template<typename T> virtual const cms::Destination*
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSDestination () const throw (cms::CMSException) [inline,
virtual]`

Gets the Destination for this **Message** (p. 2145), returns a.

Returns:

Destination object

Exceptions:

CMSException

6.47.2.13 `template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSExpiration () const throw (cms::CMSException) [inline,
virtual]`

Gets the Expiration Time for this **Message** (p. 2145).

Returns:

time value

Exceptions:

CMSException

6.47.2.14 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSMessageID () const throw (cms::CMSException) [inline,
virtual]`

Gets the CMS **Message** (p. 2145) Id for this **Message** (p. 2145).

Returns:

time value

Exceptions:

CMSException

6.47.2.15 `template<typename T> virtual int
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSPriority () const throw (cms::CMSException) [inline,
virtual]`

Gets the Priority Value for this **Message** (p. 2145).

Returns:

priority value

Exceptions:

CMSException

6.47.2.16 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSRedelivered () const throw (cms::CMSException) [inline,
virtual]`

Gets the Redelivered Flag for this **Message** (p. 2145).

Returns:

redelivered value

Exceptions:

CMSException

6.47.2.17 `template<typename T> virtual const cms::Destination*
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSReplyTo () const throw (cms::CMSException) [inline,
virtual]`

Gets the CMS Reply To Address for this **Message** (p. 2145).

Returns:

Reply To Value

Exceptions:

CMSException

6.47.2.18 `template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSTimestamp () const throw (cms::CMSException) [inline,
virtual]`

Gets the Time Stamp for this **Message** (p. 2145).

Returns:

time stamp value

Exceptions:

CMSEException

6.47.2.19 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSType () const throw (cms::CMSEException) [inline,
virtual]`

Gets the CMS Message (p. 2145) Type for this Message (p. 2145).

Returns:

type value

Exceptions:

CMSEException

6.47.2.20 `template<typename T> virtual double
activemq::commands::ActiveMQMessageTemplate< T
>::getDoubleProperty (const std::string & name) const throw (
cms::MessageFormatException, cms::CMSEException) [inline, virtual]`

Gets a double property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.47.2.21 `template<typename T> virtual float
activemq::commands::ActiveMQMessageTemplate< T
>::getFloatProperty (const std::string & name) const throw (
cms::MessageFormatException, cms::CMSEException) [inline, virtual]`

Gets a float property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.22  template<typename T> virtual int
            activemq::commands::ActiveMQMessageTemplate< T
            >::getIntProperty (const std::string & name) const throw (
            cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a int property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.23  template<typename T> virtual long long
            activemq::commands::ActiveMQMessageTemplate< T
            >::getLongProperty (const std::string & name) const throw (
            cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a long property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.


```
6.47.2.24  template<typename T> virtual std::vector<std::string>
           activemq::commands::ActiveMQMessageTemplate< T
           >::getPropertyNames () const throw ( cms::CMSEException ) [inline,
           virtual]
```

Retrieves the property names.

Returns:

The complete set of property names currently in this message.

```
6.47.2.25  template<typename T> virtual short
           activemq::commands::ActiveMQMessageTemplate< T
           >::getShortProperty (const std::string & name) const throw (
           cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a short property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.26  template<typename T> virtual std::string
           activemq::commands::ActiveMQMessageTemplate< T
           >::getStringProperty (const std::string & name) const throw (
           cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a string property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.47.2.27 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::onSend () [inline, virtual]`

Resets the **Message** (p. 2145) to a Read-Only state.

Reimplemented from **activemq::commands::Message** (p. 2156).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 202), and **activemq::commands::ActiveMQStreamMessage** (p. 468).

6.47.2.28 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::propertyExists (const std::string & name) const throw (cms::CMSException) [inline, virtual]`

Indicates whether or not a given property exists.

Parameters:

name The name of the property to look up.

Returns:

True if the property exists in this message.

6.47.2.29 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setBooleanProperty (const std::string & name, bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [inline, virtual]`

Sets a boolean property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.30 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setByteProperty (const std::string & name, unsigned char value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[inline, virtual]`

Sets a byte property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.31 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSCorrelationID (const std::string & correlationId) throw (
cms::CMSException) [inline, virtual]`

Sets the Correlation Id used by this message.

Parameters:

correlationId - String representing the correlation id.

Exceptions:

CMSException

6.47.2.32 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSDeliveryMode (int mode) throw (cms::CMSException)
[inline, virtual]`

Sets the DeliveryMode for this message.

Parameters:

mode - DeliveryMode enumerated value.

Exceptions:

CMSException

6.47.2.33 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSDestination (const cms::Destination * destination) throw (
cms::CMSException) [inline, virtual]`

Sets the Destination for this message.

Parameters:

destination - Destination Object

Exceptions:

CMSException

6.47.2.34 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSExpiration (long long expireTime) throw (
cms::CMSException) [inline, virtual]`

Sets the Expiration Time for this message.

Parameters:

expireTime - time value

Exceptions:

CMSException

6.47.2.35 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSMessageID (const std::string &id AMQCPP_UNUSED)
throw (cms::CMSException) [inline, virtual]`

Sets the CMS Message (p. 2145) Id for this message.

Parameters:

id - time value

Exceptions:

CMSException

6.47.2.36 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSPriority (int priority) throw (cms::CMSException) [inline,
virtual]`

Sets the Priority Value for this message.

Parameters:

priority - priority value for this message

Exceptions:

CMSException

6.47.2.37 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSRedelivered (bool redelivered AMQCPP_UNUSED) throw (
cms::CMSException) [inline, virtual]`

Sets the Redelivered Flag for this message.

Parameters:

redelivered - boolean redelivered value

Exceptions:

CMSException

```
6.47.2.38  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSReplyTo (const cms::Destination * destination) throw (
            cms::CMSException ) [inline, virtual]
```

Sets the CMS Reply To Address for this message.

Parameters:

destination Pointer to the CMS Destination that is the Reply-To value.

Exceptions:

CMSException

```
6.47.2.39  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSTimestamp (long long timeStamp) throw (
            cms::CMSException ) [inline, virtual]
```

Sets the Time Stamp for this message.

Parameters:

timeStamp - integer time stamp value

Exceptions:

CMSException

```
6.47.2.40  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSType (const std::string & type) throw ( cms::CMSException )
            [inline, virtual]
```

Sets the CMS **Message** (p. 2145) Type for this message.

Parameters:

type - message type value string

Exceptions:

CMSException

6.47.2.41 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setDoubleProperty (const std::string & name, double value) throw (
cms::MessageNotWriteableException, cms::CMSEException) [inline,
virtual]`

Sets a double property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.42 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setFloatProperty (const std::string & name, float value) throw (
cms::MessageNotWriteableException, cms::CMSEException) [inline,
virtual]`

Sets a float property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.43 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setIntProperty (const std::string & name, int value) throw (
cms::MessageNotWriteableException, cms::CMSEException) [inline,
virtual]`

Sets a int property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.44 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setLongProperty (const std::string & name, long long value) throw (
cms::MessageNotWriteableException, cms::CMSException) [inline,
virtual]`

Sets a long property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.45 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setShortProperty (const std::string & name, short value) throw (
cms::MessageNotWriteableException, cms::CMSException) [inline,
virtual]`

Sets a short property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.46 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setStringProperty (const std::string & name, const std::string &
value) throw (cms::MessageNotWriteableException, cms::CMSException
) [inline, virtual]`

Sets a string property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessageTemplate.h`

6.48 activemq::commands::ActiveMQObjectMessage Class Reference

#include <src/main/activemq/commands/ActiveMQObjectMessage.h> Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual **~ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **ActiveMQObjectMessage * cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*

- virtual **cms::Message * clone** () const

Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQOBJECTMESSAGE** = 26

6.48.1 Constructor & Destructor Documentation

6.48.1.1 `activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage()`

6.48.1.2 `virtual activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage()` [inline, virtual]

6.48.2 Member Function Documentation

6.48.2.1 `virtual cms::Message* activemq::commands::ActiveMQObjectMessage::clone()` const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements `cms::Message` (p. 2168).

6.48.2.2 `virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure()` const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2149).

6.48.2.3 `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2150).

6.48.2.4 `virtual bool activemq::commands::ActiveMQObjectMessage::equals(const DataStructure * value)` const [virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 369).

6.48.2.5 `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from `activemq::commands::Message` (p. 2152).

6.48.2.6 `virtual std::string activemq::commands::ActiveMQObjectMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2159).

6.48.3 Field Documentation

6.48.3.1 `const unsigned char activemq::commands::ActiveMQObjectMessage::ID_-ACTIVEMQOBJECTMESSAGE = 26 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQObjectMessage.h`

6.49 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 386).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.49.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 386).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.49.2 Constructor & Destructor Documentation

6.49.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

6.49.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

6.49.3 Member Function Documentation

6.49.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.49.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.49.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.49.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.49.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2317).

6.49

activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller

Class Reference

391

```
6.49.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller`
(p. 2318).

```
6.49.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnma
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller`
(p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h`

6.50 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 390).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 390).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.50.2 Constructor & Destructor Documentation

6.50.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` `[inline]`

6.50.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` `[inline, virtual]`

6.50.3 Member Function Documentation

6.50.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.50.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.50.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.50.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.50.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2327).

```

6.50.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

```

6.50.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h`

6.51 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 394).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.51.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 394).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.51.2 Constructor & Destructor Documentation

6.51.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

6.51.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

6.51.3 Member Function Documentation

6.51.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.51.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.51.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure&, OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.51.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.51.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2312).

```

6.51.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

```

6.51.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h`

6.52 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 398).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.52.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 398).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.52.2 Constructor & Destructor Documentation

6.52.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

6.52.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

6.52.3 Member Function Documentation

6.52.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.52.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.52.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.52.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.52.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2322).

```

6.52.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

```

6.52.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h`

6.53 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 402).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.53.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 402).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.53.2 Constructor & Destructor Documentation

6.53.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller()` [inline]

6.53.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller()` [inline, virtual]

6.53.3 Member Function Documentation

6.53.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObject(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.53.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.53.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::marshal(const commands::DataStructure&, const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.53.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.53.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2307).

```

6.53.3.6 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal
        (OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions:

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

```

6.53.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnma
        (OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
        * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h`

6.54 activemq::core::ActiveMQProducer Class Reference

#include <src/main/activemq/core/ActiveMQProducer.h> Inheritance diagram for activemq::core::ActiveMQProducer:

Public Member Functions

- **ActiveMQProducer** (const **Pointer**< **commands::ProducerInfo** > &producerInfo, const **Pointer**< **cms::Destination** > &destination, **ActiveMQSession** *session)
*Constructor, creates an instance of an **ActiveMQProducer** (p. 406).*
- virtual ~**ActiveMQProducer** ()
- virtual void **close** () throw (cms::CMSException)
Closes the Consumer.
- virtual void **send** (cms::Message *message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (cms::Message *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const cms::Destination *destination, cms::Message *message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const cms::Destination *destination, cms::Message *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **setDeliveryMode** (int mode) throw (cms::CMSException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const throw (cms::CMSException)
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value) throw (cms::CMSException)
Sets if Message Ids are disabled for this Producer.

- virtual bool **getDisableMessageID** () const throw (cms::CMSEException)
Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSEException)
Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSEException)
Gets if Message Time Stamps are disabled for this Producer.
- virtual void **setPriority** (int priority) throw (cms::CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const throw (cms::CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time) throw (cms::CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const throw (cms::CMSEException)
Gets the Time to Live that this producer sends messages with.
- virtual void **setSendTimeout** (long long time) throw (cms::CMSEException)
Sets the Send Timeout that this Producers sends messages with.
- virtual long long **getSendTimeout** () const throw (cms::CMSEException)
Gets the Send Timeout that this producer sends messages with.
- bool **isClosed** () const
- const **commands::ProducerInfo** & **getProducerInfo** () const
Retries this object ProducerInfo pointer.
- **commands::ProducerId** & **getProducerId** () const
Retries this object ProducerId or NULL if closed.
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)
Handles the work of Processing a ProducerAck Command from the Broker.

6.54.1 Constructor & Destructor Documentation

- 6.54.1.1** **activemq::core::ActiveMQProducer::ActiveMQProducer** (const **Pointer**< **commands::ProducerInfo** > & *producerInfo*, const **Pointer**< **cms::Destination** > & *destination*, **ActiveMQSession** * *session*)

Constructor, creates an instance of an **ActiveMQProducer** (p. 406).

Parameters:

- producerInfo*** Pointer to a ProducerInfo command which identifies this producer.
- destination*** The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.
- session*** The Session which is the parent of this Producer.

6.54.1.2 `virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer ()`
[virtual]

6.54.2 Member Function Documentation

6.54.2.1 `virtual void activemq::core::ActiveMQProducer::close () throw (`
`cms::CMSEException)` [virtual]

Closes the Consumer. This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions:

CMSEException

Implements **cms::Closeable** (p.1021).

6.54.2.2 `virtual int activemq::core::ActiveMQProducer::getDeliveryMode () const`
`throw (cms::CMSEException)` [inline, virtual]

Gets the delivery mode for this Producer.

Returns:

The DeliveryMode

Implements **cms::MessageProducer** (p.2333).

6.54.2.3 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID ()`
`const throw (cms::CMSEException)` [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns:

a boolean indicating state enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p.2334).

6.54.2.4 `virtual bool ac-`
`tivemq::core::ActiveMQProducer::getDisableMessageTimeStamp () const`
`throw (cms::CMSEException)` [inline, virtual]

Gets if Message Time Stamps are disabled for this Producer.

Returns:

boolean indicating state of enable / disable (true / false)

Implements **cms::MessageProducer** (p.2334).

6.54.2.5 `virtual int activemq::core::ActiveMQProducer::getPriority () const throw (cms::CMSException) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns:

int based priority level

Implements `cms::MessageProducer` (p. 2334).

6.54.2.6 `commands::ProducerId& activemq::core::ActiveMQProducer::getProducerId () const [inline]`

Retries this object ProducerId or NULL if closed.

Returns:

ProducerId Reference

6.54.2.7 `const commands::ProducerInfo& activemq::core::ActiveMQProducer::getProducerInfo () const [inline]`

Retries this object ProducerInfo pointer.

Returns:

ProducerInfo Reference

6.54.2.8 `virtual long long activemq::core::ActiveMQProducer::getSendTimeout () const throw (cms::CMSException) [inline, virtual]`

Gets the Send Timeout that this producer sends messages with.

Returns:

The default send timeout value in milliseconds.

6.54.2.9 `virtual long long activemq::core::ActiveMQProducer::getTimeToLive () const throw (cms::CMSException) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns:

The default time to live value in milliseconds.

Implements `cms::MessageProducer` (p. 2335).

6.54.2.10 `bool activemq::core::ActiveMQProducer::isClosed () const [inline]`

Returns:

true if this Producer has been closed.

6.54.2.11 `virtual void activemq::core::ActiveMQProducer::onProducerAck (const commands::ProducerAck & ack) [virtual]`

Handles the work of Processing a ProducerAck Command from the Broker.

Parameters:

ack - The ProducerAck message received from the Broker.

6.54.2.12 `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p.2335).

6.54.2.13 `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

destination The destination on which to send the message

message the message to be sent.

Exceptions:

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2335).

```
6.54.2.14 virtual void activemq::core::ActiveMQProducer::send (cms::Message
* message, int deliveryMode, int priority, long long timeToLive)
throw ( cms::CMSException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [virtual]
```

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2336).

```
6.54.2.15 virtual void activemq::core::ActiveMQProducer::send (cms::Message *
message) throw ( cms::CMSException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [virtual]
```

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

message - The message to be sent.

Exceptions:

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2336).

6.54.2.16 `virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int mode) throw (cms::CMSEException) [inline, virtual]`

Sets the delivery mode for this Producer.

Parameters:

mode - The DeliveryMode to use for Message sends.

Implements **cms::MessageProducer** (p. 2337).

6.54.2.17 `virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool value) throw (cms::CMSEException) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2337).

6.54.2.18 `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool value) throw (cms::CMSEException) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2337).

6.54.2.19 `virtual void activemq::core::ActiveMQProducer::setPriority (int priority) throw (cms::CMSEException) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters:

priority int value for Priority level

Implements **cms::MessageProducer** (p. 2338).

6.54.2.20 virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long *time*) throw (cms::CMSEException) [inline, virtual]

Sets the Send Timeout that this Producers sends messages with.

Parameters:

time The new default send timeout value in milliseconds.

6.54.2.21 virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long *time*) throw (cms::CMSEException) [inline, virtual]

Sets the Time to Live that this Producers sends messages with.

Parameters:

time The new default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2338).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

6.55 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p.2657) object.

#include <src/main/activemq/util/ActiveMQProperties.h> Inheritance diagram for activemq::util::ActiveMQProperties:

Public Member Functions

- virtual **~ActiveMQProperties** ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperties** (decaf::util::Properties &props)
- virtual bool **isEmpty** () const
Returns true if the properties object is empty.
- virtual const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- virtual void **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- virtual bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- virtual void **remove** (const std::string &name)
Removes the property with the given name.
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an arrayay.
- virtual void **copy** (const CMSProperties *source)
Copies the contents of the given properties object to this one.
- virtual CMSProperties * **clone** () const
Clones this object.
- virtual void **clear** ()
Clears all properties from the map.
- virtual std::string **toString** () const
Formats the contents of the Properties Object into a string that can be logged, etc.

6.55.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2657) object.

Since:

2.0

6.55.2 Constructor & Destructor Documentation

6.55.2.1 `virtual activemq::util::ActiveMQProperties::~~ActiveMQProperties ()` [virtual]

6.55.3 Member Function Documentation

6.55.3.1 `virtual void activemq::util::ActiveMQProperties::clear ()` [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 1037).

6.55.3.2 `virtual CMSProperties* activemq::util::ActiveMQProperties::clone ()`
`const` [virtual]

Clones this object.

Returns:

a replica of this object.

Implements **cms::CMSProperties** (p. 1037).

6.55.3.3 `virtual void activemq::util::ActiveMQProperties::copy (const`
`CMSProperties * source)` [virtual]

Copies the contents of the given properties object to this one.

Parameters:

source The source properties object.

6.55.3.4 `virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties () const` [inline, virtual]

6.55.3.5 `virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ()` [inline, virtual]

6.55.3.6 `virtual std::string activemq::util::ActiveMQProperties::getProperty (const std::string & name, const std::string & defaultValue) const` [inline, virtual]

Looks up the value for the given property.

Parameters:

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns:

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implements `cms::CMSProperties` (p. 1037).

6.55.3.7 `virtual const char* activemq::util::ActiveMQProperties::getProperty (const std::string & name) const` [inline, virtual]

Looks up the value for the given property.

Parameters:

name The name of the property to be looked up.

Returns:

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements `cms::CMSProperties` (p. 1038).

6.55.3.8 `virtual bool activemq::util::ActiveMQProperties::hasProperty (const std::string & name) const` [inline, virtual]

Check to see if the Property exists in the set.

Parameters:

name - property name to check for in this properties set.

Returns:

true if property exists, false otherwise.

Implements `cms::CMSProperties` (p. 1038).

6.55.3.9 virtual bool activemq::util::ActiveMQProperties::isEmpty () const
[inline, virtual]

Returns true if the properties object is empty.

Returns:

true if empty

Implements **cms::CMSProperties** (p. 1038).

6.55.3.10 virtual void activemq::util::ActiveMQProperties::remove (const std::string & name) [inline, virtual]

Removes the property with the given name.

Parameters:

name the name of the property to remove.

Implements **cms::CMSProperties** (p. 1038).

6.55.3.11 virtual void activemq::util::ActiveMQProperties::setProperties (decaf::util::Properties & props) [inline, virtual]**6.55.3.12 virtual void activemq::util::ActiveMQProperties::setProperty (const std::string & name, const std::string & value)** [inline, virtual]

Sets the value for a given property. If the property already exists, overwrites the value.

Parameters:

name The name of the value to be written.

value The value to be written.

Implements **cms::CMSProperties** (p. 1039).

6.55.3.13 virtual std::vector< std::pair< std::string, std::string > > activemq::util::ActiveMQProperties::toArray () const [inline, virtual]

Method that serializes the contents of the property map to an array.

Returns:

list of pairs where the first is the name and the second is the value.

Implements **cms::CMSProperties** (p. 1039).

6.55.3.14 virtual std::string activemq::util::ActiveMQProperties::toString () const
[inline, virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns:

string value of this object.

Implements **cms::CMSProperties** (p. 1039).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ActiveMQProperties.h`

6.56 activemq::commands::ActiveMQQueue Class Reference

#include <src/main/activemq/commands/ActiveMQQueue.h> Inheritance diagram for activemq::commands::ActiveMQQueue:

Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual ~**ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual **ActiveMQQueue** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination * **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual cms::Destination * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const cms::Destination &source)
Copies the contents of the given Destinastion object to this one.
- virtual const cms::CMSProperties & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getQueueName** () const throw (cms::CMSException)
Gets the name of this queue.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQQUEUE** = 100

6.56.1 Constructor & Destructor Documentation

6.56.1.1 `activemq::commands::ActiveMQQueue::ActiveMQQueue ()`

6.56.1.2 `activemq::commands::ActiveMQQueue::ActiveMQQueue (const std::string & name)`

6.56.1.3 `virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue ()`
[inline, virtual]

6.56.2 Member Function Documentation

6.56.2.1 `virtual cms::Destination* activemq::commands::ActiveMQQueue::clone ()`
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1481).

6.56.2.2 `virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 276).

6.56.2.3 `virtual void activemq::commands::ActiveMQQueue::copy (const cms::Destination & source)` [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters:

source The source Destination object.

Implements **cms::Destination** (p. 1481).

6.56.2.4 `virtual void activemq::commands::ActiveMQQueue::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 277).

6.56.2.5 `virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 278).

6.56.2.6 `virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination () const [inline, virtual]`

Returns:

the `cms::Destination` (p. 1480) interface pointer that the objects that derive from this class implement.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 278).

6.56.2.7 `virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements `cms::Destination` (p. 1481).

6.56.2.8 `virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType () const [virtual]`

Get the `DataStructure` (p. 1461) Type as defined in `CommandTypes.h`.

Returns:

The type of the data structure

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 278).

6.56.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements `activemq::commands::ActiveMQDestination` (p. 279).

References `cms::Destination::QUEUE`.

6.56.2.10 `virtual std::string activemq::commands::ActiveMQQueue::getQueueName () const throw (cms::CMSEException) [inline, virtual]`

Gets the name of this queue.

Returns:

The queue name.

Implements `cms::Queue` (p. 2674).

6.56.2.11 `virtual std::string activemq::commands::ActiveMQQueue::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 282).

6.56.3 Field Documentation

6.56.3.1 `const unsigned char activemq::commands::ActiveMQQueue::ID _ - ACTIVEMQQUEUE = 100 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQQueue.h`

6.57 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 423).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.57.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 423). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.57.2 Constructor & Destructor Documentation

6.57.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

6.57.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

6.57.3 Member Function Documentation

6.57.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.57.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.57.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 286).

6.57.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 286).

6.57.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 287).

```

6.57.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 287).

```

6.57.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 288).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h`

6.58 activemq:wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 427).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.58.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 427). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.58.2 Constructor & Destructor Documentation

6.58.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

6.58.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

6.58.3 Member Function Documentation

6.58.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.58.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.58.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 290).

6.58.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 290).

6.58.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 291).

```

6.58.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 291).

```

6.58.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 292).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h`

6.59 activemq:wireformat::openwire::marshal:v4::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 431).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v4::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.59.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 431). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.59.2 Constructor & Destructor Documentation

6.59.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

6.59.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

6.59.3 Member Function Documentation

6.59.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.59.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.59.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 294).

6.59.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 294).

6.59.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 295).

```

6.59.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 295).

```

6.59.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 296).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h`

6.60 activemq:wireformat::openwire::marshal:v5::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 435).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v5::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.60.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 435). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.60.2 Constructor & Destructor Documentation

6.60.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

6.60.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

6.60.3 Member Function Documentation

6.60.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.60.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.60.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 298).

6.60.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 298).

6.60.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 299).

```

6.60.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal2
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 299).

```

6.60.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 300).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h`

6.61 activemq:wireformat::openwire::marshal:v2::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 439).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h>Inheritance diagram for activemq:wireformat::openwire::marshal:v2::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.61.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 439). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.61.2 Constructor & Destructor Documentation

6.61.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller()` [inline]

6.61.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller()` [inline, virtual]

6.61.3 Member Function Documentation

6.61.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.61.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.61.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 302).

6.61.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 302).

6.61.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 303).

```

6.61.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 303).

```

6.61.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
    * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 304).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h`

6.62 activemq::core::ActiveMQSession Class Reference

#include <src/main/activemq/core/ActiveMQSession.h> Inheritance diagram for activemq::core::ActiveMQSession:

Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** *connection)
- virtual **~ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)
Redispatches the given set of unconsumed messages to the consumers.
- void **start** ()
Stops asynchronous message delivery.
- void **stop** ()
Starts asynchronous message delivery.
- bool **isStarted** () const
Indicates whether or not the session is currently in the started state.
- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)
Fires the given exception to the exception listener of the connection.
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Dispatches a message to a particular consumer.
- virtual void **close** () throw (**cms::CMSEException**)
Closes this session as well as any active child consumers or producers.
- virtual void **commit** () throw (**cms::CMSEException**)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** () throw (**cms::CMSEException**)
Rollsback all messages done in this transaction and releases any locks currently held.
- virtual void **recover** () throw (**cms::CMSEException**)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination) throw (**cms::CMSEException**)

Creates a MessageConsumer for the specified destination.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector) throw (cms::CMSEException)

Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)

Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSEException)

Creates a durable subscriber to the specified topic, using a message selector.

- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination) throw (cms::CMSEException)

Creates a MessageProducer to send messages to the specified destination.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue) throw (cms::CMSEException)

Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector) throw (cms::CMSEException)

Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSEException)

Creates a queue identity given a Queue name.

- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSEException)

Creates a topic identity given a Queue name.

- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSEException)

Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSEException)

Creates a TemporaryTopic object.

- virtual **cms::Message** * **createMessage** () throw (cms::CMSEException)

Creates a new Message.

- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSEException)

Creates a BytesMessage.

- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize) throw (cms::CMSEException)
Creates a BytesMessage and sets the pay-load to the passed value.
- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSEException)
Creates a new StreamMessage.
- virtual **cms::TextMessage** * **createTextMessage** () throw (cms::CMSEException)
Creates a new TextMessage.
- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text) throw (cms::CMSEException)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage** * **createMapMessage** () throw (cms::CMSEException)
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw (cms::CMSEException)
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const throw (cms::CMSEException)
Gets if the Sessions is a Transacted Session.
- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSEException)
Unsubscribes a durable subscription that has been created by a client.
- void **send** (cms::Message *message, **ActiveMQProducer** *producer, **util::Usage** *usage) throw (cms::CMSEException)
Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.
- **cms::ExceptionListener** * **getExceptionListener** ()
This method gets any registered exception listener of this sessions connection and returns it.
- const **commands::SessionInfo** & **getSessionInfo** () const
Gets the Session Information object for this session, if the session is closed than this method throws an exception.
- const **commands::SessionId** & **getSessionId** () const
Gets the Session Id object for this session, if the session is closed than this method throws an exception.
- **ActiveMQConnection** * **getConnection** () const
*Gets the **ActiveMQConnection** (p. 233) that is associated with this session.*
- long long **getLastDeliveredSequenceId** () const
Gets the currently set Last Delivered Sequence Id.
- void **setLastDeliveredSequenceId** (long long value)

Sets the value of the Last Delivered Sequence Id.

- void **oneway** (**Pointer**< **commands::Command** > command) throw (**activemq::exceptions::ActiveMQException**)

Sends a oneway message.

- void **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0) throw (**activemq::exceptions::ActiveMQException**)

Sends a synchronous request and returns the response from the broker.

- void **disposeOf** (**decaf::lang::Pointer**< **commands::ConsumerId** > id, long long lastDeliveredSequenceId) throw (**activemq::exceptions::ActiveMQException**)

Dispose of a Consumer from this session.

- void **disposeOf** (**decaf::lang::Pointer**< **commands::ProducerId** > id) throw (**activemq::exceptions::ActiveMQException**)

Dispose of a Producer from this session.

- void **doStartTransaction** () throw (**exceptions::ActiveMQException**)

Starts if not already start a Transaction for this Session.

- void **acknowledge** ()

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

- void **deliverAcks** ()

Request that this Session inform all of its consumers to deliver their pending acks.

- void **clearMessagesInProgress** ()

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

- void **wakeup** ()

Causes the Session to wakeup its executor and ensure all messages are dispatched.

Friends

- class **ActiveMQSessionExecutor**

6.62.1 Constructor & Destructor Documentation

6.62.1.1 `activemq::core::ActiveMQSession::ActiveMQSession (const Pointer< commands::SessionInfo > & sessionInfo, cms::Session::AcknowledgeMode ackMode, const decaf::util::Properties & properties, ActiveMQConnection * connection)`

6.62.1.2 `virtual activemq::core::ActiveMQSession::~~ActiveMQSession ()` [virtual]

6.62.2 Member Function Documentation

6.62.2.1 `void activemq::core::ActiveMQSession::acknowledge ()`

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

6.62.2.2 `void activemq::core::ActiveMQSession::clearMessagesInProgress ()`

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.62.2.3 `virtual void activemq::core::ActiveMQSession::close () throw (cms::CMSEException)` [virtual]

Closes this session as well as any active child consumers or producers.

Exceptions:

CMSEException

Implements `cms::Session` (p. 2842).

6.62.2.4 `virtual void activemq::core::ActiveMQSession::commit () throw (cms::CMSEException)` [virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSEException

Implements `cms::Session` (p. 2843).

6.62.2.5 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw (cms::CMSEException)` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2843).

6.62.2.6 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue) throw (cms::CMSEException) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2843).

6.62.2.7 `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage (const unsigned char * bytes, std::size_t bytesSize) throw (cms::CMSEException) [virtual]`

Creates a BytesMessage and sets the pay-load to the passed value.

Parameters:

bytes - an array of bytes to set in the message

bytesSize - the size of the bytes array, or number of bytes to use

Returns:

a newly created BytesMessage.

Exceptions:

CMSEException

Implements **cms::Session** (p. 2844).

6.62.2.8 `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage () throw (cms::CMSException) [virtual]`

Creates a BytesMessage.

Returns:

a newly created BytesMessage.

Exceptions:

CMSException

Implements `cms::Session` (p. 2844).

6.62.2.9 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSException) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination - The Destination that this consumer receiving messages for.

selector - The Message Selector string to use for this destination

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions:

CMSException

Implements `cms::Session` (p. 2844).

6.62.2.10 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw (cms::CMSException) [virtual]`

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination - The Destination that this consumer receiving messages for.

selector - The Message Selector string to use for this destination

Exceptions:

CMSException

Implements `cms::Session` (p. 2845).

6.62.2.11 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination) throw (cms::CMSException) [virtual]`

Creates a MessageConsumer for the specified destination.

Parameters:

destination - The Destination that this consumer receiving messages for.

Exceptions:

CMSException

Implements `cms::Session` (p. 2845).

6.62.2.12 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw (cms::CMSException) [virtual]`

Creates a durable subscriber to the specified topic, using a message selector.

Parameters:

destination - the topic to subscribe to

name - The name used to identify the subscription

selector - only messages matching the selector are received

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions:

CMSException

Implements `cms::Session` (p. 2846).

6.62.2.13 `virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage () throw (cms::CMSException) [virtual]`

Creates a new MapMessage.

Returns:

a newly created MapMessage.

Exceptions:

CMSException

Implements `cms::Session` (p. 2846).

6.62.2.14 `virtual cms::Message* activemq::core::ActiveMQSession::createMessage
() throw (cms::CMSException) [virtual]`

Creates a new Message.

Exceptions:

CMSException

Implements **cms::Session** (p. 2847).

6.62.2.15 `virtual cms::MessageProducer* ac-
tivismq::core::ActiveMQSession::createProducer (const
cms::Destination * destination) throw (cms::CMSException) [virtual]`

Creates a MessageProducer to send messages to the specified destination.

Parameters:

destination - the Destination to publish on

Exceptions:

CMSException

Implements **cms::Session** (p. 2847).

6.62.2.16 `virtual cms::Queue* activemq::core::ActiveMQSession::createQueue
(const std::string & queueName) throw (cms::CMSException)
[virtual]`

Creates a queue identity given a Queue name.

Parameters:

queueName - the name of the new Queue

Exceptions:

CMSException

Implements **cms::Session** (p. 2847).

6.62.2.17 `virtual cms::StreamMessage* ac-
tivismq::core::ActiveMQSession::createStreamMessage ()
throw (cms::CMSException) [virtual]`

Creates a new StreamMessage.

Returns:

a newly created StreamMessage.

Exceptions:

CMSException

Implements **cms::Session** (p. 2848).

6.62.2.18 `virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue () throw (cms::CMSException) [virtual]`

Creates a TemporaryQueue object.

Exceptions:

CMSException

Implements **cms::Session** (p. 2848).

6.62.2.19 `virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic () throw (cms::CMSException) [virtual]`

Creates a TemporaryTopic object.

Exceptions:

CMSException

Implements **cms::Session** (p. 2848).

6.62.2.20 `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage (const std::string & text) throw (cms::CMSException) [virtual]`

Creates a new TextMessage and set the text to the value given.

Parameters:

text - The initial text for the message

Returns:

a newly created TextMessage with the given Text set in the Message body.

Exceptions:

CMSException

Implements **cms::Session** (p. 2848).

6.62.2.21 `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage () throw (cms::CMSException) [virtual]`

Creates a new TextMessage.

Returns:

a newly created `TextMessage`.

Exceptions:

CMSEException

Implements `cms::Session` (p. 2849).

6.62.2.22 `virtual cms::Topic* activemq::core::ActiveMQSession::createTopic (const std::string & topicName) throw (cms::CMSEException) [virtual]`

Creates a topic identity given a Queue name.

Parameters:

topicName - the name of the new Topic

Exceptions:

CMSEException

Implements `cms::Session` (p. 2849).

6.62.2.23 `void activemq::core::ActiveMQSession::deliverAcks ()`

Request that this Session inform all of its consumers to deliver their pending acks.

6.62.2.24 `virtual void activemq::core::ActiveMQSession::dispatch (const Pointer< MessageDispatch > & message) [virtual]`

Dispatches a message to a particular consumer.

Parameters:

message - the message to be dispatched

Implements `activemq::core::Dispatcher` (p. 1536).

6.62.2.25 `void activemq::core::ActiveMQSession::disposeOf (decaf::lang::Pointer< commands::ProducerId > id) throw (activemq::exceptions::ActiveMQException)`

Dispose of a Producer from this session. Removes it from the Connection and clean up any resources associated with it.

Parameters:

id - the Id of the Producer to dispose.

Exceptions:

ActiveMQException if an internal error occurs.

6.62.2.26 `void activemq::core::ActiveMQSession::disposeOf (decaf::lang::Pointer< commands::ConsumerId > id, long long lastDeliveredSequenceId) throw (activemq::exceptions::ActiveMQException)`

Dispose of a Consumer from this session. Removes it from the Connection and clean up any resources associated with it.

Parameters:

id The Id of the Consumer to dispose.

lastDeliveredSequenceId The Broker Sequence Id of the last message the Consumer delivered.

Exceptions:

ActiveMQException if an internal error occurs.

6.62.2.27 `void activemq::core::ActiveMQSession::doStartTransaction () throw (exceptions::ActiveMQException)`

Starts if not already start a Transaction for this Session. If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

Exceptions:

ActiveMQException if this is not a Transacted Session.

6.62.2.28 `void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & ex)`

Fires the given exception to the exception listener of the connection.

6.62.2.29 `virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode () const throw (cms::CMSException) [virtual]`

Returns the acknowledgment mode of the session.

Returns:

the Sessions Acknowledge Mode

Implements `cms::Session` (p. 2849).

6.62.2.30 `ActiveMQConnection* activemq::core::ActiveMQSession::getConnection () const [inline]`

Gets the `ActiveMQConnection` (p. 233) that is associated with this session.

6.62.2.31 `cms::ExceptionListener* activemq::core::ActiveMQSession::getExceptionListener()`

This method gets any registered exception listener of this sessions connection and returns it. Mainly intended for use by the objects that this session creates so that they can notify the client of exceptions that occur in the context of another thread.

Returns:

`cms::ExceptionListener` (p. 1581) pointer or NULL

6.62.2.32 `long long activemq::core::ActiveMQSession::getLastDeliveredSequenceId() const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns:

long long containing the sequence id of the last delivered Message.

6.62.2.33 `const commands::SessionId& activemq::core::ActiveMQSession::getSessionId() const [inline]`

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

Returns:

SessionId Reference

6.62.2.34 `const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo() const [inline]`

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

Returns:

SessionInfo Reference

6.62.2.35 `bool activemq::core::ActiveMQSession::isAutoAcknowledge() const [inline]`

References `cms::Session::AUTO_ACKNOWLEDGE`.

6.62.2.36 `bool activemq::core::ActiveMQSession::isClientAcknowledge() const [inline]`

References `cms::Session::CLIENT_ACKNOWLEDGE`.

6.62.2.37 `bool activemq::core::ActiveMQSession::isDupsOkAcknowledge () const [inline]`

References `cms::Session::DUPS_OK_ACKNOWLEDGE`.

6.62.2.38 `bool activemq::core::ActiveMQSession::isIndividualAcknowledge () const [inline]`

References `cms::Session::INDIVIDUAL_ACKNOWLEDGE`.

6.62.2.39 `bool activemq::core::ActiveMQSession::isStarted () const`

Indicates whether or not the session is currently in the started state.

6.62.2.40 `virtual bool activemq::core::ActiveMQSession::isTransacted () const throw (cms::CMSEException) [virtual]`

Gets if the Sessions is a Transacted Session.

Returns:

transacted true - false.

Implements `cms::Session` (p. 2850).

6.62.2.41 `void activemq::core::ActiveMQSession::oneway (Pointer< commands::Command > command) throw (activemq::exceptions::ActiveMQException)`

Sends a oneway message.

Parameters:

command The message to send.

Exceptions:

ActiveMQException if not currently connected, or if the operation fails for any reason.

6.62.2.42 `virtual void activemq::core::ActiveMQSession::recover () throw (cms::CMSEException) [virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"

- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions:

CMSEException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

Implements **cms::Session** (p. 2850).

6.62.2.43 void activemq::core::ActiveMQSession::redispatch (MessageDispatchChannel & *unconsumedMessages*)

Redispatches the given set of unconsumed messages to the consumers.

Parameters:

unconsumedMessages - unconsumed messages to be redelivered.

6.62.2.44 virtual void activemq::core::ActiveMQSession::rollback () throw (cms::CMSEException) [virtual]

Rollsback all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSEException

Implements **cms::Session** (p. 2850).

6.62.2.45 void activemq::core::ActiveMQSession::send (cms::Message * *message*, ActiveMQProducer * *producer*, util::Usage * *usage*) throw (cms::CMSEException)

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection. Asynchronous sends will be chosen if at all possible.

Parameters:

message The message to send to the broker.

producer The sending Producer

usage Pointer to a Usage tracker which if set will be increased by the size of the given message.

Exceptions:

CMSEException

6.62.2.46 `void activemq::core::ActiveMQSession::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters:

value The new value to assign to the Last Delivered Sequence Id property.

6.62.2.47 `void activemq::core::ActiveMQSession::start ()`

Stops asynchronous message delivery.

6.62.2.48 `void activemq::core::ActiveMQSession::stop ()`

Starts asynchronous message delivery.

6.62.2.49 `void activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw (activemq::exceptions::ActiveMQException)`

Sends a synchronous request and returns the response from the broker. Converts any error responses into an exception.

Parameters:

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions:

ActiveMQException thrown if an error response was received from the broker, or if any other error occurred.

6.62.2.50 `virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & name) throw (cms::CMSException) [virtual]`

Unsubscribes a durable subscription that has been created by a client. This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters:

name the name used to identify this subscription

Exceptions:

CMSException

Implements `cms::Session` (p. 2851).

6.62.2.51 void activemq::core::ActiveMQSession::wakeup ()

Causes the Session to wakeup its executor and ensure all messages are dispatched.

6.62.3 Friends And Related Function Documentation

6.62.3.1 friend class ActiveMQSessionExecutor [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSession.h**

6.63 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

#include <src/main/activemq/core/ActiveMQSessionExecutor.h> Inheritance diagram for activemq::core::ActiveMQSessionExecutor:

Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** *session)
Creates an un-started executor for the given session.
- virtual **~ActiveMQSessionExecutor** ()
*Calls **stop()** (p. 463) then **clear()** (p. 461).*
- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.
- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.
- virtual void **clearMessagesInProgress** ()
Removes all messages in the Dispatch Channel so that non are delivered.
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()
wakeup this executor and dispatch any pending messages.
- virtual void **start** ()
Starts the dispatching.
- virtual void **stop** ()
Stops dispatching.
- virtual void **close** ()
Terminates the dispatching thread.
- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()
Removes all queued messages and destroys them.
- virtual bool **iterate** ()
*Iterates on the **MessageDispatchChannel** (p. 2224) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

6.63.1 Detailed Description

Delegate dispatcher for a single session. Contains a thread to provide for asynchronous dispatching.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 `activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor (ActiveMQSession * session)`

Creates an un-started executor for the given session.

6.63.2.2 `virtual activemq::core::ActiveMQSessionExecutor::~~ActiveMQSessionExecutor () [virtual]`

Calls `stop()` (p. 463) then `clear()` (p. 461).

6.63.3 Member Function Documentation

6.63.3.1 `virtual void activemq::core::ActiveMQSessionExecutor::clear () [inline, virtual]`

Removes all queued messages and destroys them.

6.63.3.2 `virtual void ac- tivemq::core::ActiveMQSessionExecutor::clearMessagesInProgress () [inline, virtual]`

Removes all messages in the Dispatch Channel so that non are delivered.

6.63.3.3 `virtual void activemq::core::ActiveMQSessionExecutor::close () [inline, virtual]`

Terminates the dispatching thread. Once this is called, the executor is no longer usable.

6.63.3.4 `virtual void activemq::core::ActiveMQSessionExecutor::execute (const Pointer< MessageDispatch > & data) [virtual]`

Executes the dispatch. Adds the given data to the end of the queue.

Parameters:

data - the data to be dispatched.

6.63.3.5 `virtual void activemq::core::ActiveMQSessionExecutor::executeFirst (const Pointer< MessageDispatch > & data) [virtual]`

Executes the dispatch. Adds the given data to the beginning of the queue.

Parameters:

data - the data to be dispatched.

6.63.3.6 `std::vector< Pointer<MessageDispatch> > activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages ()`
[inline]

Returns:

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

6.63.3.7 `virtual bool activemq::core::ActiveMQSessionExecutor::hasUnconsumedMessages ()`
`const` [inline, virtual]

Returns:

true if there are any pending messages in the dispatch channel.

6.63.3.8 `virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ()`
[inline, virtual]

Returns:

true if there are no messages in the Dispatch Channel.

6.63.3.9 `virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ()` `const`
[inline, virtual]

Returns:

true indicates if the executor is started

6.63.3.10 `virtual bool activemq::core::ActiveMQSessionExecutor::iterate ()`
[virtual]

Iterates on the **MessageDispatchChannel** (p. 2224) sending all pending messages to the Consumers they are destined for.

Returns:

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 3148).

6.63.3.11 `virtual void activemq::core::ActiveMQSessionExecutor::start ()`
[virtual]

Starts the dispatching.

6.63.3.12 virtual void activemq::core::ActiveMQSessionExecutor::stop () [virtual]

Stops dispatching.

6.63.3.13 virtual void activemq::core::ActiveMQSessionExecutor::wakeup () [virtual]

wakeup this executor and dispatch any pending messages.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSessionExecutor.h**

6.64 activemq::commands::ActiveMQStreamMessage Class Reference

#include <src/main/activemq/commands/ActiveMQStreamMessage.h> Inheritance diagram for activemq::commands::ActiveMQStreamMessage:

Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQStreamMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()
Store the Data that was written to the stream before a send.
- virtual **cms::StreamMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **reset** () throw (cms::CMSException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value) throw (cms::MessageNotWriteableException, cms::CMSException)
Writes a boolean to the Stream message stream as a 1-byte value.

- virtual unsigned char **readByte** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Stream message stream.
- virtual void **writeByte** (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the Stream message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the Stream message stream.
- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Char from the Stream message stream.
- virtual void **writeChar** (char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a char to the Stream message stream as a 1-byte value.
- virtual float **readFloat** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit float from the Stream message stream.
- virtual void **writeFloat** (float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a float to the Stream message stream as a 4 byte value.
- virtual double **readDouble** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the Stream message stream as a 8 byte value.

- virtual short **readShort** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Stream message stream.

- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the Stream message stream as a 2 byte value.

- virtual int **readInt** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the Stream message stream as a 4 byte value.

- virtual long long **readLong** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Stream message stream.

- virtual void **writeLong** (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the Stream message stream as a 8 byte value.

- virtual std::string **readString** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Stream message stream.

- virtual void **writeString** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Stream message stream.

Static Public Attributes

- static const unsigned char `ID_ACTIVEMQSTREAMMESSAGE` = 27

6.64.1 Constructor & Destructor Documentation

6.64.1.1 `activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage()`

6.64.1.2 `virtual activemq::commands::ActiveMQStreamMessage::~~ActiveMQStreamMessage()` [virtual]

6.64.2 Member Function Documentation

6.64.2.1 `virtual void activemq::commands::ActiveMQStreamMessage::clearBody () throw (cms::CMSException)` [virtual]

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 368).

6.64.2.2 `virtual cms::StreamMessage* activemq::commands::ActiveMQStreamMessage::clone () const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements `cms::Message` (p. 2168).

6.64.2.3 `virtual ActiveMQStreamMessage* activemq::commands::ActiveMQStreamMessage::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2149).

6.64.2.4 `virtual void activemq::commands::ActiveMQStreamMessage::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2150).

6.64.2.5 `virtual bool activemq::commands::ActiveMQStreamMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 369).

6.64.2.6 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new `DataStructure` (p. 1461) type copy.

Reimplemented from `activemq::commands::Message` (p. 2152).

6.64.2.7 `virtual void activemq::commands::ActiveMQStreamMessage::onSend () [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 376).

6.64.2.8 `virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException) [virtual]`

Reads a Boolean from the Stream message stream.

Returns:

boolean value from stream

Exceptions:

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3084).

6.64.2.9 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Byte from the Stream message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3084).

6.64.2.10 `virtual std::size_t activemq::commands::ActiveMQStreamMessage::readBytes (unsigned char * buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a portion of the Stream message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `CMSException` is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to `value.length`

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3085).

6.64.2.11 `virtual std::size_t activemq::commands::ActiveMQStreamMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)` [virtual]

Reads a byte array from the Stream message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3085).

6.64.2.12 `virtual char activemq::commands::ActiveMQStreamMessage::readChar () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)` [virtual]

Reads a Char from the Stream message stream.

Returns:

char value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3086).

6.64.2.13 virtual double `activemq::commands::ActiveMQStreamMessage::readDouble`
 () const throw (`cms::MessageEOFException`,
`cms::MessageFormatException`, `cms::MessageNotReadableException`,
`cms::CMSException`) [virtual]

Reads a 64 bit double from the Stream message stream.

Returns:

double value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3086).

6.64.2.14 virtual float `activemq::commands::ActiveMQStreamMessage::readFloat`
 () const throw (`cms::MessageEOFException`,
`cms::MessageFormatException`, `cms::MessageNotReadableException`,
`cms::CMSException`) [virtual]

Reads a 32 bit float from the Stream message stream.

Returns:

double value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3087).

6.64.2.15 `virtual int activemq::commands::ActiveMQStreamMessage::readInt
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a 32 bit signed integer from the Stream message stream.

Returns:

int value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3087).

6.64.2.16 `virtual long long ac-
tivismq::commands::ActiveMQStreamMessage::readLong
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a 64 bit long from the Stream message stream.

Returns:

long long value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3087).

6.64.2.17 `virtual short activemq::commands::ActiveMQStreamMessage::readShort
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a 16 bit signed short from the Stream message stream.

Returns:

short value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3088).

6.64.2.18 `virtual std::string activemq::commands::ActiveMQStreamMessage::readString
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads an ASCII String from the Stream message stream.

Returns:

String from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3088).

6.64.2.19 `virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3089).

6.64.2.20 virtual void activemq::commands::ActiveMQStreamMessage::reset ()
throw (cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions:

CMSException

6.64.2.21 virtual std::string activemq::commands::ActiveMQStreamMessage::toString ()
const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2159).

6.64.2.22 virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool *value*)
throw (cms::MessageNotWriteableException, cms::CMSException)
[virtual]

Writes a boolean to the Stream message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value boolean to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3089).

6.64.2.23 virtual void activemq::commands::ActiveMQStreamMessage::writeByte
(unsigned char *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a byte to the Stream message stream as a 1-byte value.

Parameters:

value byte to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3089).

6.64.2.24 `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const unsigned char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a portion of a byte array to the Stream message stream. size as the number of bytes to write.

Parameters:

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3090).

6.64.2.25 `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters:

value bytes to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3090).

6.64.2.26 `virtual void activemq::commands::ActiveMQStreamMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a char to the Stream message stream as a 1-byte value.

Parameters:

value char to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3090).

6.64.2.27 virtual void activemq::commands::ActiveMQStreamMessage::writeDouble
(double *value*) throw (cms::MessageNotWriteableException,
cms::CMSEException) [virtual]

Writes a double to the Stream message stream as a 8 byte value.

Parameters:

value double to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3091).

6.64.2.28 virtual void activemq::commands::ActiveMQStreamMessage::writeFloat
(float *value*) throw (cms::MessageNotWriteableException,
cms::CMSEException) [virtual]

Writes a float to the Stream message stream as a 4 byte value.

Parameters:

value float to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3091).

6.64.2.29 virtual void activemq::commands::ActiveMQStreamMessage::writeInt
(int *value*) throw (cms::MessageNotWriteableException,
cms::CMSEException) [virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters:

value signed int to write to the stream

Exceptions:

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3091).

6.64.2.30 virtual void activemq::commands::ActiveMQStreamMessage::writeLong
(long long *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a long long to the Stream message stream as a 8 byte value.

Parameters:

value signed long long to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements cms::StreamMessage (p. 3092).

6.64.2.31 virtual void activemq::commands::ActiveMQStreamMessage::writeShort
(short *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters:

value signed short to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements cms::StreamMessage (p. 3092).

6.64.2.32 virtual void activemq::commands::ActiveMQStreamMessage::writeString
(const std::string & *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes an ASCII String to the Stream message stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements cms::StreamMessage (p. 3092).

6.64.2.33 `virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters:

value unsigned short to write to the stream

Exceptions:

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

Implements `cms::StreamMessage` (p. 3093).

6.64.3 Field Documentation

6.64.3.1 `const unsigned char activemq::commands::ActiveMQStreamMessage::ID_ - ACTIVEMQSTREAMMESSAGE = 27 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQStreamMessage.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 479).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h>
Inheritance diagram for activemq:wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller:
```

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.65.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 479).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.65.2 Constructor & Destructor Documentation

6.65.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

6.65.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

6.65.3 Member Function Documentation

6.65.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.65.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.65.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.65

activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller

Class Reference

483

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2316).

6.65.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2316).

6.65.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2317).

6.65.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

6.65.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h`

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 483).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h>
Inheritance diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller:
```

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.66.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 483).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

6.66.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

6.66.3 Member Function Documentation

6.66.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.66.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.66.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.66

activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller

Class Reference

487

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2326).

6.66.3.4 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2326).

6.66.3.5 virtual int ac-

```
tivemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2327).

6.66.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

6.66.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 487).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h>
Inheritance diagram for activemq:wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller:
```

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.67.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 487).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.67.2 Constructor & Destructor Documentation

6.67.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

6.67.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

6.67.3 Member Function Documentation

6.67.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.67.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.67.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.67

activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller

Class Reference

491

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2311).

6.67.3.4 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2311).

6.67.3.5 virtual int ac-

```
ativemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2312).

6.67.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

6.67.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 491).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h>
Inheritance diagram for activemq:wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller:
```

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.68.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 491).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.68.2 Constructor & Destructor Documentation

6.68.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

6.68.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

6.68.3 Member Function Documentation

6.68.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.68.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.68.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.68

activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

Class Reference

495

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2321).

6.68.3.4 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2321).

6.68.3.5 virtual int ac-

```
tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2322).

6.68.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

6.68.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 495).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h>
Inheritance diagram for activemq:wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller:
```

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.69.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 495).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.69.2 Constructor & Destructor Documentation

6.69.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

6.69.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

6.69.3 Member Function Documentation

6.69.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.69.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.69.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.69

activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller

Class Reference

499

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2306).

6.69.3.4 virtual void ac-

```
tivemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataInputStream * dataIn) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2306).

6.69.3.5 virtual int ac-

```
tivemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2307).

6.69.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

6.69.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h`

6.70 activemq::commands::ActiveMQTempDestination Class Reference

#include <src/main/activemq/commands/ActiveMQTempDestination.h> Inheritance diagram for activemq::commands::ActiveMQTempDestination:

Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempDestination** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual void **close** () throw (cms::CMSException)
Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.
- void **setConnection** (core::ActiveMQConnection *connection)
Sets the Parent Connection that is notified when this destination is destroyed.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPDESTINATION** = 0

Protected Attributes

- core::ActiveMQConnection * **connection**
Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.70.1 Constructor & Destructor Documentation

- 6.70.1.1** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination()`
- 6.70.1.2** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination(const std::string & name)`
- 6.70.1.3** `virtual activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination()` [virtual]

6.70.2 Member Function Documentation

- 6.70.2.1** `virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataStructure()` `const` [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 276).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 524), and `activemq::commands::ActiveMQTempTopic` (p. 549).

- 6.70.2.2** `virtual void activemq::commands::ActiveMQTempDestination::close()` `throw (cms::CMSException)` [virtual]

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker. This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws `cms::CMSException` (p. 1031)

Implements `cms::Closeable` (p. 1021).

- 6.70.2.3** `virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure(const DataStructure * src)` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 277).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 525), and `activemq::commands::ActiveMQTempTopic` (p. 550).

References `activemq::commands::ActiveMQDestination::copyDataStructure()`.

**6.70.2.4 virtual bool activemq::commands::ActiveMQTempDestination::equals
(const DataStructure * *value*) const [inline, virtual]**

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 278).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 525), and **activemq::commands::ActiveMQTempTopic** (p. 550).

References **activemq::commands::ActiveMQDestination::equals()**.

**6.70.2.5 virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType ()
const [virtual]**

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 278).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 526), and **activemq::commands::ActiveMQTempTopic** (p. 551).

**6.70.2.6 void activemq::commands::ActiveMQTempDestination::setConnection
(core::ActiveMQConnection * *connection*) [inline]**

Sets the Parent Connection that is notified when this destination is destroyed.

Parameters:

connection - The parent connection.

**6.70.2.7 virtual std::string activemq::commands::ActiveMQTempDestination::toString ()
const [virtual]**

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 282).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 526), and **activemq::commands::ActiveMQTempTopic** (p. 551).

6.70.3 Field Documentation

6.70.3.1 `core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection`
[protected]

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.70.3.2 `const unsigned char activemq::commands::ActiveMQTempDestination::ID -`
`ACTIVEMQTEMPDESTINATION = 0` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 503).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
diagram for activemq:wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller:
```

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.71.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 503).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.71.2 Constructor & Destructor Documentation

6.71.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.71.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.71.3 Member Function Documentation

6.71.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 286).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 529), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 554).

6.71.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

6.71

activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller

Class Reference

507

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 286).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 530), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 555).

6.71.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * *wireFormat*, commands::DataStructure *
***dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)**
[virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 287).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 530), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 555).

6.71.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 287).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 530), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 555).

6.71.3.5 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 288).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 531), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 556).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 507).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
diagram for activemq:wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller:
```

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.72.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 507).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.72.2 Constructor & Destructor Documentation

6.72.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.72.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.72.3 Member Function Documentation

6.72.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 290).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 533), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 558).

6.72.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

6.72

activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller

Class Reference

511

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 290).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 534), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 559).

6.72.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 291).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 534), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 559).

6.72.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 291).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 534), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 559).

6.72.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 292).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 535), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 560).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h`

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 511).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
diagram for activemq:wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller:
```

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.73.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 511).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.73.2 Constructor & Destructor Documentation

6.73.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.73.2.2 `virtual ~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.73.3 Member Function Documentation

6.73.3.1 `virtual void activate(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 294).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 537), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 562).

6.73.3.2 `virtual void deactivate(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

6.73

activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller

Class Reference

515

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 294).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 538), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 563).

6.73.3.3 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMar
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)
[virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 295).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 538), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 563).

6.73.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 295).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 538), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 563).

6.73.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 296).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 539), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 564).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h`

6.74

activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller

Class Reference

6.74 ~~activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller~~⁵¹⁷

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 515).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller:
```

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.74.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 515).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.74.2 Constructor & Destructor Documentation

6.74.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.74.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.74.3 Member Function Documentation

6.74.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 298).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 541), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 566).

6.74.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

6.74

activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller

Class Reference

519

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 298).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 542), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 567).

6.74.3.3 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 299).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 542), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 567).

6.74.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 299).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 542), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 567).

6.74.3.5 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 300).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 543), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 568).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h`

6.75

activemq:wireformat::openwire::marshal:v2::ActiveMQTempDestinationMarshaller

Class Reference

6.75 ~~activemq:wireformat::openwire::marshal:v2::ActiveMQTempDestinationMarshaller~~⁵²¹

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 519).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
diagram for activemq:wireformat::openwire::marshal:v2::ActiveMQTempDestinationMarshaller:
```

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.75.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 519).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.75.2 Constructor & Destructor Documentation

6.75.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.75.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.75.3 Member Function Documentation

6.75.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 302).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 545), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 570).

6.75.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

6.75

activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

Class Reference

523

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 302).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 546), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 571).

6.75.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * *wireFormat*, commands::DataStructure *
***dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)**
[virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 303).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 546), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 571).

6.75.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 303).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 546), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 571).

6.75.3.5 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 304).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 547), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 572).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h`

6.76 activemq::commands::ActiveMQTempQueue Class Reference

#include <src/main/activemq/commands/ActiveMQTempQueue.h> Inheritance diagram for activemq::commands::ActiveMQTempQueue:

Public Member Functions

- **ActiveMQTempQueue** ()
- **ActiveMQTempQueue** (const std::string &name)
- virtual ~**ActiveMQTempQueue** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempQueue** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Converts the Destination Name into a String.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destinastion object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getQueueName** () const throw (cms::CMSException)
Gets the name of this queue.
- virtual void **destroy** () throw (cms::CMSException)
Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPQUEUE** = 102

6.76.1 Constructor & Destructor Documentation

6.76.1.1 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ()`

6.76.1.2 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue (const std::string & name)`

6.76.1.3 `virtual
activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue ()
[virtual]`

6.76.2 Member Function Documentation

6.76.2.1 `virtual cms::Destination* ac-
tivemq::commands::ActiveMQTempQueue::clone () const
[inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1481).

6.76.2.2 `virtual ActiveMQTempQueue* ac-
tivemq::commands::ActiveMQTempQueue::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 500).

6.76.2.3 `virtual void activemq::commands::ActiveMQTempQueue::copy (const
cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters:

source The source Destination object.

Implements **cms::Destination** (p. 1481).

6.76.2.4 `virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 500).

6.76.2.5 `virtual void activemq::commands::ActiveMQTempQueue::destroy () throw (cms::CMSException) [virtual]`

Destroy's the Temp Destination at the Broker.

Exceptions:

CMSException

Implements `cms::TemporaryQueue` (p. 3166).

6.76.2.6 `virtual bool activemq::commands::ActiveMQTempQueue::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 501).

6.76.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination () const [inline, virtual]`

Returns:

the `cms::Destination` (p. 1480) interface pointer that the objects that derive from this class implement.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 278).

6.76.2.8 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempQueue::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1481).

6.76.2.9 `virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 501).

6.76.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 279).

References cms::Destination::TEMPORARY_QUEUE.

6.76.2.11 `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName () const throw (cms::CMSException) [inline, virtual]`

Gets the name of this queue.

Returns:

The queue name.

Implements **cms::TemporaryQueue** (p. 3167).

6.76.2.12 `virtual std::string activemq::commands::ActiveMQTempQueue::toString () const [virtual]`

Converts the Destination Name into a String.

Returns:

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 501).

6.76.3 Field Documentation

6.76.3.1 `const unsigned char activemq::commands::ActiveMQTempQueue::ID_ - ACTIVEMQTEMPQUEUE = 102` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempQueue.h`

6.77 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 528).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.77.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 528).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.77.2 Constructor & Destructor Documentation

6.77.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` `[inline]`

6.77.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` `[inline, virtual]`

6.77.3 Member Function Documentation

6.77.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.77.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.77.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 504).

6.77.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 504).

6.77.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 505).

6.77.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 505).

6.77.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 506).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h`

6.78 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 532).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.78.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 532).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.78.2 Constructor & Destructor Documentation

6.78.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` `[inline]`

6.78.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` `[inline, virtual]`

6.78.3 Member Function Documentation

6.78.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.78.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.78.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 508).

6.78.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 508).

6.78.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509).

6.78.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509).

6.78.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 510).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h`

6.79 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 536).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.79.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 536).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.79.2 Constructor & Destructor Documentation

6.79.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

6.79.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

6.79.3 Member Function Documentation

6.79.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.79.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.79.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 512).

6.79.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 512).

6.79.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 513).

6.79.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 513).

6.79.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 514).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h`

6.80 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 540).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.80.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 540).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.80.2 Constructor & Destructor Documentation

6.80.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` [inline]

6.80.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` [inline, virtual]

6.80.3 Member Function Documentation

6.80.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.80.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.80.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 516).

6.80.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 516).

6.80.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 517).

6.80.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 517).

6.80.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 518).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h`

6.81 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 544).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.81.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 544).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.81.2 Constructor & Destructor Documentation

6.81.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller()` `[inline]`

6.81.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller()` `[inline, virtual]`

6.81.3 Member Function Documentation

6.81.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.81.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.81.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 520).

6.81.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 520).

6.81.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 521).

6.81.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 521).

6.81.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 522).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h`

6.82 activemq::commands::ActiveMQTempTopic Class Reference

#include <src/main/activemq/commands/ActiveMQTempTopic.h> Inheritance diagram for activemq::commands::ActiveMQTempTopic:

Public Member Functions

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual ~**ActiveMQTempTopic** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempTopic** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Converts the Destination Name into a String.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destinastion object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getTopicName** () const throw (cms::CMSException)
Gets the name of this topic.
- virtual void **destroy** () throw (cms::CMSException)
Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPTOPIC** = 103

6.82.1 Constructor & Destructor Documentation

6.82.1.1 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ()`

6.82.1.2 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & name)`

6.82.1.3 `virtual
activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic ()
[virtual]`

6.82.2 Member Function Documentation

6.82.2.1 `virtual cms::Destination* ac-
tivemq::commands::ActiveMQTempTopic::clone () const
[inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1481).

6.82.2.2 `virtual ActiveMQTempTopic* ac-
tivemq::commands::ActiveMQTempTopic::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 500).

6.82.2.3 `virtual void activemq::commands::ActiveMQTempTopic::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters:

source The source Destination object.

Implements **cms::Destination** (p. 1481).

6.82.2.4 `virtual void activemq::commands::ActiveMQTempTopic::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 500).

6.82.2.5 `virtual void activemq::commands::ActiveMQTempTopic::destroy () throw (cms::CMSException) [virtual]`

Destroy's the Temp Destination at the Broker.

Exceptions:

CMSException

Implements `cms::TemporaryTopic` (p. 3168).

6.82.2.6 `virtual bool activemq::commands::ActiveMQTempTopic::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 501).

6.82.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination () const [inline, virtual]`

Returns:

the `cms::Destination` (p. 1480) interface pointer that the objects that derive from this class implement.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 278).

6.82.2.8 `virtual const cms::CMSPProperties& activemq::commands::ActiveMQTempTopic::getCMSPProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1481).

6.82.2.9 `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 501).

6.82.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 279).

References cms::Destination::TEMPORARY_TOPIC.

6.82.2.11 `virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName () const throw (cms::CMSException) [inline, virtual]`

Gets the name of this topic.

Returns:

The topic name.

Implements **cms::TemporaryTopic** (p. 3169).

6.82.2.12 `virtual std::string activemq::commands::ActiveMQTempTopic::toString () const [virtual]`

Converts the Destination Name into a String.

Returns:

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 501).

6.82.3 Field Documentation

6.82.3.1 `const unsigned char activemq::commands::ActiveMQTempTopic::ID - ACTIVEMQTEMPTOPIC = 103` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempTopic.h`

6.83 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 553).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.83.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 553). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.83.2 Constructor & Destructor Documentation

6.83.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` [inline]

6.83.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` [inline, virtual]

6.83.3 Member Function Documentation

6.83.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.83.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.83.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 504).

6.83.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 504).

6.83.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 505).

6.83.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 505).

6.83.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 506).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h`

6.84 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 557).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.84.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 557). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.84.2 Constructor & Destructor Documentation

6.84.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` [inline]

6.84.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` [inline, virtual]

6.84.3 Member Function Documentation

6.84.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.84.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.84.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 508).

6.84.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 508).

6.84.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509).

```

6.84.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509).

```

6.84.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 510).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h`

6.85 activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 561).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.85.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 561). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.85.2 Constructor & Destructor Documentation

6.85.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` [inline]

6.85.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` [inline, virtual]

6.85.3 Member Function Documentation

6.85.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.85.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.85.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 512).

6.85.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 512).

6.85.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 513).

```

6.85.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal2
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 513).

```

6.85.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 514).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h`

6.86 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 565).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.86.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 565). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.86.2 Constructor & Destructor Documentation

6.86.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` [inline]

6.86.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` [inline, virtual]

6.86.3 Member Function Documentation

6.86.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.86.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.86.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 516).

6.86.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 516).

6.86.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 517).

6.86.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal2(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 517).

6.86.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 518).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h`

6.87 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 569).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.87.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 569). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.87.2 Constructor & Destructor Documentation

6.87.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller()` [inline]

6.87.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller()` [inline, virtual]

6.87.3 Member Function Documentation

6.87.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.87.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.87.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 520).

6.87.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 520).

6.87.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 521).

```

6.87.3.6 virtual void ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2
          (OpenWireFormat * wireFormat, commands::DataStructure
          * dataStructure, decaf::io::DataOutputStream * dataOut,
          utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 521).

```

6.87.3.7 virtual void ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarshal
          (OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
          * bs) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 522).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h`

6.88 activemq::commands::ActiveMQTextMessage Class Reference

#include <src/main/activemq/commands/ActiveMQTextMessage.h> Inheritance diagram for activemq::commands::ActiveMQTextMessage:

Public Member Functions

- **ActiveMQTextMessage** ()
- virtual **~ActiveMQTextMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQTextMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **beforeMarshal** (wireformat::WireFormat *wireFormat) throw (decaf::io::IOException)
*Performs any cleanup or other tasks that must be done before the **Message** (p. 2145) is marshalled to its binary WireFormat version.*
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.
- virtual cms::TextMessage * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual std::string **getText** () const throw (cms::CMSException)
Gets the message character buffer.
- virtual void **setText** (const char *msg) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets the message contents, does not take ownership of the passed char, but copies it instead.*

- virtual void **setText** (const std::string &msg) throw (cms::MessageNotWriteableException, cms::CMSException)

Sets the message contents.

Data Fields

- std::auto_ptr< std::string > **text**

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEXTMESSAGE** = 28

6.88.1 Constructor & Destructor Documentation

6.88.1.1 `activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ()`

6.88.1.2 `virtual
activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage ()
[virtual]`

6.88.2 Member Function Documentation

6.88.2.1 `virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal
(wireformat::WireFormat * wireFormat) throw (decaf::io::IOException)
[virtual]`

Performs any cleanup or other tasks that must be done before the **Message** (p. 2145) is marshalled to its binary WireFormat version.

Parameters:

wireFormat the WireFormat instance that is marshalling this message.

Implements `activemq::wireformat::MarshalAware` (p. 2119).

6.88.2.2 `virtual void activemq::commands::ActiveMQTextMessage::clearBody ()
throw (cms::CMSException) [virtual]`

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 368).

6.88.2.3 `virtual cms::TextMessage* ac-
tivemq::commands::ActiveMQTextMessage::clone () const
[inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 2168).

6.88.2.4 virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2149).

6.88.2.5 virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure (const DataStructure * src) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2150).

6.88.2.6 virtual bool activemq::commands::ActiveMQTextMessage::equals (const DataStructure * value) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 369).

6.88.2.7 virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from **activemq::commands::Message** (p. 2152).

6.88.2.8 `virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize
() const [virtual]`

Returns the Size of this message in Bytes.

Returns:

number of bytes this message equates to.

Reimplemented from `activemq::commands::Message` (p. 2154).

6.88.2.9 `virtual std::string activemq::commands::ActiveMQTextMessage::getText
() const throw (cms::CMSException) [virtual]`

Gets the message character buffer.

Returns:

The message character buffer.

Exceptions:

CMSException - if an internal error occurs.

Implements `cms::TextMessage` (p. 3170).

6.88.2.10 `virtual void activemq::commands::ActiveMQTextMessage::setText
(const std::string & msg) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]`

Sets the message contents.

Parameters:

msg The message buffer.

Exceptions:

CMSException - if an internal error occurs.

MessageNotWriteableException - if the message is in read-only mode..

Implements `cms::TextMessage` (p. 3171).

6.88.2.11 `virtual void activemq::commands::ActiveMQTextMessage::setText
(const char * msg) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]`

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters:

msg The message buffer.

Exceptions:

CMSException - if an internal error occurs.

MessageNotWriteableException - if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3171).

**6.88.2.12 virtual std::string activemq::commands::ActiveMQTextMessage::toString()
() const [virtual]**

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2159).

6.88.3 Field Documentation**6.88.3.1 const unsigned char activemq::commands::ActiveMQTextMessage::ID_ -
ACTIVEMQTEXTMESSAGE = 28 [static]****6.88.3.2 std::auto_ptr<std::string> ac-
tivemq::commands::ActiveMQTextMessage::text
[mutable]**

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTextMessage.h**

6.89 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 578).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.89.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 578).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.89.2 Constructor & Destructor Documentation

6.89.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

6.89.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

6.89.3 Member Function Documentation

6.89.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.89.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.89.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.89.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2316).

6.89.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2317).

6.89.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

6.89.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2318).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h`

6.90 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 582).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.90.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 582).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.90.2 Constructor & Destructor Documentation

6.90.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` `[inline]`

6.90.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` `[inline, virtual]`

6.90.3 Member Function Documentation

6.90.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject(const std::string& name) const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.90.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.90.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarshal(const commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.90.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326).

6.90.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2327).

6.90.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

6.90.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h`

6.91 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 586).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.91.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 586).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.91.2 Constructor & Destructor Documentation

6.91.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

6.91.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

6.91.3 Member Function Documentation

6.91.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.91.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.91.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.91.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2311).

6.91.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2312).

6.91.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

6.91.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h`

6.92 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 590).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.92.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 590).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.92.2 Constructor & Destructor Documentation

6.92.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

6.92.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

6.92.3 Member Function Documentation

6.92.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.92.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.92.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseMarshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.92.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321).

6.92.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2322).

6.92.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

6.92.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h`

6.93 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 594).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.93.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 594).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.93.2 Constructor & Destructor Documentation

6.93.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

6.93.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

6.93.3 Member Function Documentation

6.93.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject(const std::string& name) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.93.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.93.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.93.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2306).

6.93.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2307).

6.93.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

6.93.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2308).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h`

6.94 activemq::commands::ActiveMQTopic Class Reference

#include <src/main/activemq/commands/ActiveMQTopic.h> Inheritance diagram for activemq::commands::ActiveMQTopic:

Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTopic** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination * **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual cms::Destination * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const cms::Destination &source)
Copies the contents of the given Destination object to this one.
- virtual const cms::CMSProperties & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getTopicName** () const throw (cms::CMSException)
Gets the name of this topic.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTOPIC** = 101

6.94.1 Constructor & Destructor Documentation

6.94.1.1 `activemq::commands::ActiveMQTopic::ActiveMQTopic ()`

6.94.1.2 `activemq::commands::ActiveMQTopic::ActiveMQTopic (const std::string & name)`

6.94.1.3 `virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic ()`
[virtual]

6.94.2 Member Function Documentation

6.94.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTopic::clone ()`
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1481).

6.94.2.2 `virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 276).

6.94.2.3 `virtual void activemq::commands::ActiveMQTopic::copy (const cms::Destination & source)` [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters:

source The source Destination object.

Implements **cms::Destination** (p. 1481).

6.94.2.4 `virtual void activemq::commands::ActiveMQTopic::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 277).

6.94.2.5 `virtual bool activemq::commands::ActiveMQTopic::equals (const DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 278).

References **activemq::commands::ActiveMQDestination::equals()**.

6.94.2.6 `virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination () const [inline, virtual]`

Returns:

the **cms::Destination** (p. 1480) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 278).

6.94.2.7 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties () const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1481).

6.94.2.8 `virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 278).

6.94.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType () const [inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 279).

References **cms::Destination::TOPIC**.

6.94.2.10 `virtual std::string activemq::commands::ActiveMQTopic::getTopicName () const throw (cms::CMSEException) [inline, virtual]`

Gets the name of this topic.

Returns:

The topic name.

Implements **cms::Topic** (p. 3215).

6.94.2.11 `virtual std::string activemq::commands::ActiveMQTopic::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 282).

6.94.3 Field Documentation

6.94.3.1 `const unsigned char activemq::commands::ActiveMQTopic::ID_ - ACTIVEMQTOPIC = 101 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTopic.h`

6.95 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 602).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.95.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 602). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.95.2 Constructor & Destructor Documentation

6.95.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

6.95.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

6.95.3 Member Function Documentation

6.95.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.95.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.95.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 286).

6.95.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 286).

6.95.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 287).

6.95.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 287).

6.95.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 288).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h`

6.96 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 606).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.96.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 606). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.96.2 Constructor & Destructor Documentation

6.96.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` `[inline]`

6.96.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` `[inline, virtual]`

6.96.3 Member Function Documentation

6.96.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.96.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.96.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 290).

6.96.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 290).

6.96.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 291).

6.96.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 291).

6.96.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 292).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h`

6.97 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 610).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.97.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 610). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.97.2 Constructor & Destructor Documentation

6.97.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` `[inline]`

6.97.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` `[inline, virtual]`

6.97.3 Member Function Documentation

6.97.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.97.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.97.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 294).

6.97.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 294).

6.97.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 295).

6.97.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 295).

6.97.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 296).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h`

6.98 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 614).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.98.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 614). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.98.2 Constructor & Destructor Documentation

6.98.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` [inline]

6.98.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` [inline, virtual]

6.98.3 Member Function Documentation

6.98.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.98.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.98.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 298).

6.98.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 298).

6.98.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 299).

6.98.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 299).

6.98.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 300).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h`

6.99 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 618).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.99.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 618). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.99.2 Constructor & Destructor Documentation

6.99.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller()` `[inline]`

6.99.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller()` `[inline, virtual]`

6.99.3 Member Function Documentation

6.99.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.99.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.99.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 302).

6.99.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 302).

6.99.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 303).

6.99.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 303).

6.99.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 304).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h`

6.100 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransactionContext.h>
```

Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** *session, const **decaf::util::Properties** &properties)
Constructor.
- virtual **~ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Adds a **Synchronization** (p. 3137) to this Transaction.*
- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Removes a **Synchronization** (p. 3137) to this Transaction.*
- virtual void **begin** () throw (exceptions::ActiveMQException)
Begins a new transaction if one is not currently in progress.
- virtual void **commit** () throw (exceptions::ActiveMQException)
Commit the current Transaction.
- virtual void **rollback** () throw (exceptions::ActiveMQException)
Rollback the current Transaction.
- virtual const **decaf::lang::Pointer**< **commands::TransactionId** > & **getTransactionId** () const
Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.
- virtual bool **isInTransaction** () const
Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.
- virtual int **getMaximumRedeliveries** () const
- virtual long long **getRedeliveryDelay** () const

6.100.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back. The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

Configuration options

transaction.maxRedeliveryCount Max number of times a message can be re-delivered, if the session is rolled back more than this many time, the message is dropped.

transaction.redeliveryDelay Time in Milliseconds between message redelivery for rolled back transactions.

Since:

2.0

6.100.2 Constructor & Destructor Documentation

6.100.2.1 activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext (ActiveMQSession * *session*, const decaf::util::Properties & *properties*)

Constructor.

Parameters:

session The session that contains this transaction
properties Configuration parameters for this object

6.100.2.2 virtual activemq::core::ActiveMQTransactionContext::~~ActiveMQTransactionContext () [virtual]

6.100.3 Member Function Documentation

6.100.3.1 virtual void activemq::core::ActiveMQTransactionContext::addSynchronization (const Pointer< Synchronization > & *sync*) [virtual]

Adds a **Synchronization** (p. 3137) to this Transaction.

Parameters:

sync - The **Synchronization** (p. 3137) instance to add.

6.100.3.2 virtual void activemq::core::ActiveMQTransactionContext::begin () throw (exceptions::ActiveMQException) [virtual]

Begins a new transaction if one is not currently in progress.

Exceptions:

ActiveMQException

6.100.3.3 virtual void activemq::core::ActiveMQTransactionContext::commit () throw (exceptions::ActiveMQException) [virtual]

Commit the current Transaction.

Exceptions:*ActiveMQException*

6.100.3.4 `virtual int activemq::core::ActiveMQTransactionContext::getMaximumRedeliveries () const [virtual]`

Returns:

The Maximum number of time the client will attempt to redeliver a message from a rolled back transaction before marking the message as not consumed by this client.

6.100.3.5 `virtual long long activemq::core::ActiveMQTransactionContext::getRedeliveryDelay () const [virtual]`

Returns:

The time in Milliseconds that this client is configured to wait in between redelivery attempts for a Message in a rolled back transaction.

6.100.3.6 `virtual const decaf::lang::Pointer<commands::TransactionId>& activemq::core::ActiveMQTransactionContext::getTransactionId () const [virtual]`

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

Returns:

TransactionInfo

Exceptions:

InvalidStateException if a Transaction is not in progress.

6.100.3.7 `virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction () const [virtual]`

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

Returns:

true if a transaction is in progress.

6.100.3.8 `virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & sync) [virtual]`

Removes a **Synchronization** (p. 3137) to this Transaction.

Parameters:

sync - The **Synchronization** (p. 3137) instance to add.

6.100.3.9 `virtual void activemq::core::ActiveMQTransactionContext::rollback ()
throw (exceptions::ActiveMQException) [virtual]`

Rollback the current Transaction.

Exceptions:

ActiveMQException

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQTransactionContext.h`

6.101 decaf::lang::Appendable Class Reference

#include <src/main/decaf/lang/Appendable.h> Inheritance diagram for decaf::lang::Appendable:

Public Member Functions

- virtual **~Appendable** ()
- virtual **Appendable & append** (char value)=0 throw (decaf::lang::Exception)
*Appends the specified character to this **Appendable** (p. 626).*
- virtual **Appendable & append** (const **CharSequence** *csq)=0 throw (decaf::lang::Exception)
*Appends the specified character sequence to this **Appendable** (p. 626).*
- virtual **Appendable & append** (const **CharSequence** *csq, std::size_t start, std::size_t end)=0 throw (decaf::lang::Exception)
*Appends a subsequence of the specified character sequence to this **Appendable** (p. 626).*

6.101.1 Constructor & Destructor Documentation

6.101.1.1 virtual decaf::lang::Appendable::~~Appendable () [inline, virtual]

6.101.2 Member Function Documentation

6.101.2.1 virtual Appendable& decaf::lang::Appendable::append (const **CharSequence** * *csq*, std::size_t *start*, std::size_t *end*) throw (decaf::lang::Exception) [pure virtual]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 626).

Parameters:

- csq* - The character sequence from which a subsequence will be appended. If csq is NULL, then characters will be appended as if csq contained the string "null".
- start* - The index of the first character in the subsequence
- end* - The index of the character following the last character in the subsequence

Returns:

a Reference to this **Appendable** (p. 626)

Exceptions:

- Exception* (p. 1574) if an error occurs.
- IndexOutOfBoundsException* start is greater than end, or end is greater than csq.length()

Implemented in **decaf::nio::CharBuffer** (p. 1001).

6.101.2.2 `virtual Appendable& decaf::lang::Appendable::append (const CharSequence * csq) throw (decaf::lang::Exception)` [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 626).

Parameters:

csq - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

Returns:

a Reference to this **Appendable** (p. 626)

Exceptions:

Exception (p. 1574) if an error occurs.

Implemented in **decaf::nio::CharBuffer** (p. 1002).

6.101.2.3 `virtual Appendable& decaf::lang::Appendable::append (char value) throw (decaf::lang::Exception)` [pure virtual]

Appends the specified character to this **Appendable** (p. 626).

Parameters:

value - The character to append

Returns:

a Reference to this **Appendable** (p. 626)

Exceptions:

Exception (p. 1574) if an error occurs.

Implemented in **decaf::nio::CharBuffer** (p. 1002).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Appendable.h**

6.102 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

Public Member Functions

- **AprPool ()**
- **virtual ~AprPool ()**
- **apr_pool_t * getAprPool () const**
Gets the internal APR Pool.
- **void cleanup ()**
Clears data that was allocated by this pool.

Static Public Member Functions

- **static apr_pool_t * getGlobalPool ()**
Gets a pointer to the Global APR Pool used for the Application.

6.102.1 Detailed Description

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

6.102.2 Constructor & Destructor Documentation

6.102.2.1 decaf::internal::AprPool::AprPool ()

6.102.2.2 virtual decaf::internal::AprPool::~~AprPool () [virtual]

6.102.3 Member Function Documentation

6.102.3.1 void decaf::internal::AprPool::cleanup ()

Clears data that was allocated by this pool. Users should call this after getting the data from the APR functions and copying it to someplace safe.

6.102.3.2 apr_pool_t* decaf::internal::AprPool::getAprPool () const

Gets the internal APR Pool.

Returns:

the internal APR pool

6.102.3.3 static apr_pool_t* decaf::internal::AprPool::getGlobalPool () [static]

Gets a pointer to the Global APR Pool used for the Application.

Returns:

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**AprPool.h**

6.103 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Public Member Functions

- **AtomicBoolean** ()
*Creates a new **AtomicBoolean** (p. 630) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)
*Creates a new **AtomicBoolean** (p. 630) with the initial value.*
- virtual ~**AtomicBoolean** ()
- bool **get** () const
*Gets the current value of this **AtomicBoolean** (p. 630).*
- void **set** (bool newValue)
Unconditionally sets to the given value.
- bool **compareAndSet** (bool expect, bool update)
Atomically sets the value to the given updated value if the current value == the expected value.
- bool **getAndSet** (bool newValue)
Atomically sets to the given value and returns the previous value.
- std::string **toString** () const
Returns the String representation of the current value.

6.103.1 Detailed Description

A boolean value that may be updated atomically. An **AtomicBoolean** (p. 630) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ()

Creates a new **AtomicBoolean** (p. 630) whose initial value is false.

6.103.2.2 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean (bool initialValue)

Creates a new **AtomicBoolean** (p. 630) with the initial value.

Parameters:

initialValue - The initial value of this boolean.

6.103.2.3 virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean()
[inline, virtual]

6.103.3 Member Function Documentation

6.103.3.1 bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet
(bool *expect*, bool *update*)

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters:

expect - the expected value

update - the new value

Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

6.103.3.2 bool decaf::util::concurrent::atomic::AtomicBoolean::get () const
[inline]

Gets the current value of this **AtomicBoolean** (p. 630).

Returns:

the currently set value.

6.103.3.3 bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet (bool
new Value)

Atomically sets to the given value and returns the previous value.

Parameters:

new Value - the new value

Returns:

the previous value

6.103.3.4 void decaf::util::concurrent::atomic::AtomicBoolean::set (bool *new Value*)
[inline]

Unconditionally sets to the given value.

Parameters:

new Value - the new value

6.103.3.5 `std::string decaf::util::concurrent::atomic::AtomicBoolean::toString ()`
`const`

Returns the String representation of the current value.

Returns:

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicBoolean.h`

6.104 decaf::util::concurrent::atomic::AtomicInteger Class Reference

An int value that may be updated atomically.

#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h> Inheritance diagram for decaf::util::concurrent::atomic::AtomicInteger:

Public Member Functions

- **AtomicInteger** ()
*Create a new **AtomicInteger** (p. 633) with an initial value of 0.*
- **AtomicInteger** (int initialValue)
*Create a new **AtomicInteger** (p. 633) with the given initial value.*
- virtual **~AtomicInteger** ()
- int **get** () const
Gets the current value.
- void **set** (int newValue)
Sets to the given value.
- int **getAndSet** (int newValue)
Atomically sets to the given value and returns the old value.
- bool **compareAndSet** (int expect, int update)
Atomically sets the value to the given updated value if the current value == the expected value.
- int **getAndIncrement** ()
Atomically increments by one the current value.
- int **getAndDecrement** ()
Atomically decrements by one the current value.
- int **getAndAdd** (int delta)
Atomically adds the given value to the current value.
- int **incrementAndGet** ()
Atomically increments by one the current value.
- int **decrementAndGet** ()
Atomically decrements by one the current value.
- int **addAndGet** (int delta)
Atomically adds the given value to the current value.
- std::string **toString** () const

Returns the String representation of the current value.

- int **intValue** () const

Description copied from class: Number Returns the value of the specified number as an int.

- long long **longValue** () const

Description copied from class: Number Returns the value of the specified number as a long.

- float **floatValue** () const

Description copied from class: Number Returns the value of the specified number as a float.

- double **doubleValue** () const

Description copied from class: Number Returns the value of the specified number as a double.

6.104.1 Detailed Description

An int value that may be updated atomically. An **AtomicInteger** (p. 633) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ()

Create a new **AtomicInteger** (p. 633) with an initial value of 0.

6.104.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 633) with the given initial value.

Parameters:

initialValue - The initial value of this object.

6.104.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger () [inline, virtual]

6.104.3 Member Function Documentation

6.104.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet (int *delta*)

Atomically adds the given value to the current value.

Parameters:

delta - the value to add.

Returns:

the updated value.

6.104.3.2 bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet (int *expect*, int *update*)

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters:

expect - the expected value

update - the new value

Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

6.104.3.3 int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ()

Atomically decrements by one the current value.

Returns:

the updated value.

Referenced by decaf::lang::AtomicRefCounter::release().

6.104.3.4 double decaf::util::concurrent::atomic::AtomicInteger::doubleValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a double. This may involve rounding.

Returns:

the numeric value represented by this object after conversion to type double.

Implements **decaf::lang::Number** (p. 2433).

6.104.3.5 float decaf::util::concurrent::atomic::AtomicInteger::floatValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a float. This may involve rounding.

Returns:

the numeric value represented by this object after conversion to type float.

Implements **decaf::lang::Number** (p. 2433).

6.104.3.6 `int decaf::util::concurrent::atomic::AtomicInteger::get () const [inline]`

Gets the current value.

Returns:

the current value.

6.104.3.7 `int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd (int delta)`

Atomically adds the given value to the current value.

Parameters:

delta - The value to add.

Returns:

the previous value.

6.104.3.8 `int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ()`

Atomically decrements by one the current value.

Returns:

the previous value.

6.104.3.9 `int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ()`

Atomically increments by one the current value.

Returns:

the previous value.

6.104.3.10 `int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int new Value)`

Atomically sets to the given value and returns the old value.

Parameters:

new Value - the new value.

Returns:

the previous value.

6.104.3.11 int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ()

Atomically increments by one the current value.

Returns:

the updated value.

Referenced by `decaf::lang::AtomicRefCounter::AtomicRefCounter()`.

6.104.3.12 int decaf::util::concurrent::atomic::AtomicInteger::intValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as an int. This may involve rounding or truncation.

Returns:

the numeric value represented by this object after conversion to type int.

Implements **decaf::lang::Number** (p. 2433).

6.104.3.13 long long decaf::util::concurrent::atomic::AtomicInteger::longValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a long. This may involve rounding or truncation.

Returns:

the numeric value represented by this object after conversion to type long long.

Implements **decaf::lang::Number** (p. 2433).

6.104.3.14 void decaf::util::concurrent::atomic::AtomicInteger::set (int *new Value*) [inline]

Sets to the given value.

Parameters:

new Value - the new value

6.104.3.15 std::string decaf::util::concurrent::atomic::AtomicInteger::toString () const

Returns the String representation of the current value.

Returns:

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicInteger.h`

6.105 decaf::lang::AtomicRefCounter Class Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

Inherited by `decaf::lang::Pointer< activemq::threads::TaskRunner >`, `decaf::lang::Pointer< ActiveMQDestination >`, `decaf::lang::Pointer< ActiveMQTransactionContext >`, `decaf::lang::Pointer< BackupTransportPool >`, `decaf::lang::Pointer< BooleanExpression >`, `decaf::lang::Pointer< BrokerError >`, `decaf::lang::Pointer< BrokerId >`, `decaf::lang::Pointer< CloseTransportsTask >`, `decaf::lang::Pointer< cms::Destination >`, `decaf::lang::Pointer< commands::BrokerInfo >`, `decaf::lang::Pointer< commands::ConnectionInfo >`, `decaf::lang::Pointer< commands::ConsumerInfo >`, `decaf::lang::Pointer< commands::Message >`, `decaf::lang::Pointer< commands::MessageAck >`, `decaf::lang::Pointer< commands::ProducerInfo >`, `decaf::lang::Pointer< commands::SessionInfo >`, `decaf::lang::Pointer< commands::TransactionId >`, `decaf::lang::Pointer< commands::WireFormatInfo >`, `decaf::lang::Pointer< Comparator< E > >`, `decaf::lang::Pointer< CompositeTaskRunner >`, `decaf::lang::Pointer< ConnectionId >`, `decaf::lang::Pointer< ConnectionInfo >`, `decaf::lang::Pointer< ConsumerId >`, `decaf::lang::Pointer< ConsumerInfo >`, `decaf::lang::Pointer< core::ActiveMQAckHandler >`, `decaf::lang::Pointer< DataStructure >`, `decaf::lang::Pointer< decaf::lang::Runnable >`, `decaf::lang::Pointer< decaf::lang::Thread >`, `decaf::lang::Pointer< decaf::util::Properties >`, `decaf::lang::Pointer< Exception >`, `decaf::lang::Pointer< InactivityMonitorData >`, `decaf::lang::Pointer< LockHandle >`, `decaf::lang::Pointer< Message >`, `decaf::lang::Pointer< MessageAck >`, `decaf::lang::Pointer< MessageId >`, `decaf::lang::Pointer< ProducerId >`, `decaf::lang::Pointer< ProducerInfo >`, `decaf::lang::Pointer< Response >`, `decaf::lang::Pointer< ResponseBuilder >`, `decaf::lang::Pointer< SessionId >`, `decaf::lang::Pointer< SessionInfo >`, `decaf::lang::Pointer< Tracked >`, `decaf::lang::Pointer< TransactionId >`, `decaf::lang::Pointer< Transport >`, `decaf::lang::Pointer< transport::Transport >`, `decaf::lang::Pointer< TransportListener >`, `decaf::lang::Pointer< URI >`, `decaf::lang::Pointer< URIPool >`, and `decaf::lang::Pointer< wireformat::WireFormat >`.

Public Member Functions

- `AtomicRefCounter ()`
- `AtomicRefCounter (const AtomicRefCounter &other)`

Protected Member Functions

- `void swap (AtomicRefCounter &other)`

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

- `bool release ()`

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

6.105.1 Constructor & Destructor Documentation

6.105.1.1 decaf::lang::AtomicRefCounter::AtomicRefCounter () [inline]

6.105.1.2 decaf::lang::AtomicRefCounter::AtomicRefCounter (const AtomicRefCounter & *other*) [inline]

References decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet().

6.105.2 Member Function Documentation

6.105.2.1 bool decaf::lang::AtomicRefCounter::release () [inline, protected]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

Returns:

true if the count is now zero.

Reimplemented in decaf::lang::Pointer< commands::ConsumerId > (p. 2502), decaf::lang::Pointer< MessageAck > (p. 2502), decaf::lang::Pointer< BooleanExpression > (p. 2502), decaf::lang::Pointer< BrokerError > (p. 2502), decaf::lang::Pointer< Transport > (p. 2502), decaf::lang::Pointer< wireformat::WireFormat > (p. 2502), decaf::lang::Pointer< CloseTransportsTask > (p. 2502), decaf::lang::Pointer< CompositeTaskRunner > (p. 2502), decaf::lang::Pointer< commands::WireFormatInfo > (p. 2502), decaf::lang::Pointer< ActiveMQTransactionContext > (p. 2502), decaf::lang::Pointer< commands::ProducerInfo > (p. 2502), decaf::lang::Pointer< Comparator< E > > (p. 2502), decaf::lang::Pointer< BrokerId > (p. 2502), decaf::lang::Pointer< commands::SessionInfo > (p. 2502), decaf::lang::Pointer< Message > (p. 2502), decaf::lang::Pointer< URI > (p. 2502), decaf::lang::Pointer< DataStructure > (p. 2502), decaf::lang::Pointer< commands::ConnectionInfo > (p. 2502), decaf::lang::Pointer< activemq::threads::TaskRunner > (p. 2502), decaf::lang::Pointer< LockHandle > (p. 2502), decaf::lang::Pointer< ConsumerInfo > (p. 2502), decaf::lang::Pointer< ConnectionId > (p. 2502), decaf::lang::Pointer< decaf::lang::Runnable > (p. 2502), decaf::lang::Pointer< BackupTransportPool > (p. 2502), decaf::lang::Pointer< commands::BrokerInfo > (p. 2502), decaf::lang::Pointer< ProducerInfo > (p. 2502), decaf::lang::Pointer< decaf::lang::Thread > (p. 2502), decaf::lang::Pointer< MessageId > (p. 2502), decaf::lang::Pointer< Response > (p. 2502), decaf::lang::Pointer< SessionId > (p. 2502), decaf::lang::Pointer< cms::Destination > (p. 2502), decaf::lang::Pointer< TransportListener > (p. 2502), decaf::lang::Pointer< commands::TransactionId > (p. 2502), decaf::lang::Pointer< ActiveMQDestination > (p. 2502), decaf::lang::Pointer< ProducerId > (p. 2502), decaf::lang::Pointer< ResponseBuilder > (p. 2502), decaf::lang::Pointer< SessionInfo > (p. 2502), decaf::lang::Pointer< commands::Message > (p. 2502), decaf::lang::Pointer< transport::Transport > (p. 2502), decaf::lang::Pointer< InactivityMonitorData > (p. 2502), decaf::lang::Pointer< Tracked > (p. 2502), decaf::lang::Pointer< ConnectionInfo > (p. 2502), decaf::lang::Pointer< commands::MessageAck > (p. 2502), decaf::lang::Pointer< core::ActiveMQAckHandler > (p. 2502), decaf::lang::Pointer< Exception > (p. 2502), decaf::lang::Pointer< commands::ConsumerInfo > (p. 2502), decaf::lang::Pointer< ConsumerId > (p. 2502), decaf::lang::Pointer<

URIPool > (p. 2502), **decaf::lang::Pointer< decaf::util::Properties >** (p. 2502), and **decaf::lang::Pointer< TransactionId >** (p. 2502).

References `decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet()`.

6.105.2.2 `void decaf::lang::AtomicRefCounter::swap (AtomicRefCounter & other)`
[inline, protected]

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

Parameters:

other The value to swap with this one's.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.106 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

Public Member Functions

- **AtomicReference** ()
- **AtomicReference** (T *value)
- virtual ~**AtomicReference** ()
- T * **get** () const

Gets the Current Value.

- void **set** (T *newValue)

Sets the Current value of this Reference.

- bool **compareAndSet** (T *expect, T *update)

Atomically sets the value to the given updated value if the current value == the expected value.

- T * **getAndSet** (T *newValue)

Atomically sets to the given value and returns the old value.

- std::string **toString** () const

Returns the String representation of the current value.

6.106.1 Detailed Description

```
template<typename T> class decaf::util::concurrent::atomic::AtomicReference< T >
```

An Pointer reference that may be updated atomically.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 `template<typename T >
decaf::util::concurrent::atomic::AtomicReference< T
>::AtomicReference () [inline]`

6.106.2.2 `template<typename T >
decaf::util::concurrent::atomic::AtomicReference< T
>::AtomicReference (T * value) [inline]`

6.106.2.3 `template<typename T > virtual
decaf::util::concurrent::atomic::AtomicReference< T
>::~~AtomicReference () [inline, virtual]`

6.106.3 Member Function Documentation

6.106.3.1 `template<typename T > bool
decaf::util::concurrent::atomic::AtomicReference< T
>::compareAndSet (T * expect, T * update) [inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters:

expect - the expected value

update - the new value

Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

6.106.3.2 `template<typename T > T*
decaf::util::concurrent::atomic::AtomicReference< T >::get
() const [inline]`

Gets the Current Value.

Returns:

the current value of this Reference.

6.106.3.3 `template<typename T > T*
decaf::util::concurrent::atomic::AtomicReference< T
>::getAndSet (T * new Value) [inline]`

Atomically sets to the given value and returns the old value.

Parameters:

new Value- the new value

Returns:

the previous value.

6.106.3.4 `template<typename T > void
decaf::util::concurrent::atomic::AtomicReference< T >::set
(T * new Value) [inline]`

Sets the Current value of this Reference.

Parameters:

new Value The new Value of this Reference.

6.106.3.5 `template<typename T > std::string
decaf::util::concurrent::atomic::AtomicReference< T
>::toString () const [inline]`

Returns the String representation of the current value.

Returns:

string representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicReference.h`

6.107 activemq::transport::failover::BackupTransport Class Reference

#include <src/main/activemq/transport/failover/BackupTransport.h> Inheritance diagram for activemq::transport::failover::BackupTransport:

Public Member Functions

- **BackupTransport** (**BackupTransportPool** *failover)
- virtual **~BackupTransport** ()
- **decaf::net::URI** **getUri** () const
Gets the URI assigned to this Backup.
- void **setUri** (const **decaf::net::URI** &uri)
*Sets the URI assigned to this **Transport** (p. 3273).*
- const **Pointer**< **Transport** > & **getTransport** ()
Gets the currently held transport.
- void **setTransport** (const **Pointer**< **Transport** > &transport)
Sets the held transport, if not NULL then start to listen for exceptions from the held transport.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- bool **isClosed** () const
*Has the **Transport** (p. 3273) been shutdown and no longer usable.*
- void **setClosed** (bool value)
*Sets the closed flag on this **Transport** (p. 3273).*

6.107.1 Constructor & Destructor Documentation

- 6.107.1.1 **activemq::transport::failover::BackupTransport::BackupTransport** (**BackupTransportPool** * failover)
- 6.107.1.2 virtual **activemq::transport::failover::BackupTransport::~~BackupTransport** ()
[virtual]

6.107.2 Member Function Documentation

- 6.107.2.1 const **Pointer**<**Transport**>& **activemq::transport::failover::BackupTransport::getTransport** () [inline]

Gets the currently held transport.

Returns:

pointer to the held transport or NULL if not set.

6.107.2.2 decaf::net::URI activemq::transport::failover::BackupTransport::getUri () const [inline]

Gets the URI assigned to this Backup.

Returns:

the assigned URI

6.107.2.3 bool activemq::transport::failover::BackupTransport::isClosed () const [inline]

Has the **Transport** (p. 3273) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 3273)

6.107.2.4 virtual void activemq::transport::failover::BackupTransport::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command transport. The **BackupTransport** (p. 644) closes its internal **Transport** (p. 3273) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

Parameters:

ex The exception that was passed to this listener to handle.

Implements **activemq::transport::TransportListener** (p. 3290).

6.107.2.5 void activemq::transport::failover::BackupTransport::setClosed (bool *value*) [inline]

Sets the closed flag on this **Transport** (p. 3273).

Parameters:

value - true for closed.

6.107.2.6 void activemq::transport::failover::BackupTransport::setTransport (const Pointer< Transport > & *transport*) [inline]

Sets the held transport, if not NULL then start to listen for exceptions from the held transport.

Parameters:

transport The transport to hold.

6.107.2.7 `void activemq::transport::failover::BackupTransport::setUri (const decaf::net::URI & uri) [inline]`

Sets the URI assigned to this **Transport** (p. 3273).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransport.h`

6.108 activemq::transport::failover::BackupTransportPool Class Reference

#include <src/main/activemq/transport/failover/BackupTransportPool.h> Inheritance diagram for activemq::transport::failover::BackupTransportPool:

Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const
Return true if we don't currently have enough Connected Transports.
- **Pointer**< **BackupTransport** > **getBackup** ()
*Get a Connected **Transport** (p. 3273) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*
- virtual bool **iterate** ()
Connect to a Backup Broker if we haven't already connected to the max number of Backups.
- int **getBackupPoolSize** () const
Gets the Max number of Backups this Task will create.
- void **setBackupPoolSize** (int size)
Sets the Max number of Backups this Task will create.
- bool **isEnabled** () const
*Gets if the backup **Transport** (p. 3273) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*
- void **setEnabled** (bool value)
*Sets if this Backup **Transport** (p. 3273) Pool is enabled.*

Friends

- class **BackupTransport**

6.108.1 Constructor & Destructor Documentation

6.108.1.1 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.108.1.2 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (int backupPoolSize, const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.108.1.3 `virtual
activemq::transport::failover::BackupTransportPool::~~BackupTransportPool
() [virtual]`

6.108.2 Member Function Documentation

6.108.2.1 `Pointer<BackupTransport> activemq::transport::failover::BackupTransportPool::getBackup
()`

Get a Connected **Transport** (p. 3273) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

Returns:

Pointer to a Connected **Transport** (p. 3273) or NULL

6.108.2.2 `int activemq::transport::failover::BackupTransportPool::getBackupPoolSize ()
const [inline]`

Gets the Max number of Backups this Task will create.

Returns:

the max number of active BackupTransports that will be created.

6.108.2.3 `bool activemq::transport::failover::BackupTransportPool::isEnabled ()
const [inline]`

Gets if the backup **Transport** (p. 3273) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

Returns:

true if enable.

6.108.2.4 `virtual bool activemq::transport::failover::BackupTransportPool::isPending () const` [virtual]

Return true if we don't currently have enough Connected Transports.

Implements **activemq::threads::CompositeTask** (p. 1090).

6.108.2.5 `virtual bool activemq::transport::failover::BackupTransportPool::iterate ()` [virtual]

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements **activemq::threads::Task** (p. 3148).

6.108.2.6 `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize (int size)` [inline]

Sets the Max number of Backups this Task will create.

Parameters:

size - the max number of active BackupTransports that will be created.

6.108.2.7 `void activemq::transport::failover::BackupTransportPool::setEnabled (bool value)`

Sets if this Backup **Transport** (p. 3273) Pool is enabled. When not enabled no Backups are created and any that were are destroyed.

Parameters:

value - true to enable backup creation, false to disable.

6.108.3 Friends And Related Function Documentation

6.108.3.1 `friend class BackupTransport` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

6.109 activemq::commands::BaseCommand Class Reference

#include <src/main/activemq/commands/BaseCommand.h> Inheritance diagram for activemq::commands::BaseCommand:

Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)
*Sets the **Command** (p. 1063) Id of this **Message** (p. 2145).*
- virtual int **getCommandId** () const
*Gets the **Command** (p. 1063) Id of this **Message** (p. 2145).*
- virtual void **setResponseRequired** (const bool required)
*Set if this **Message** (p. 2145) requires a **Response** (p. 2781).*
- virtual bool **isResponseRequired** () const
*Is a **Response** (p. 2781) required for this **Command** (p. 1063).*
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

6.109.1 Constructor & Destructor Documentation

6.109.1.1 `activemq::commands::BaseCommand::BaseCommand ()` [inline]

6.109.1.2 `virtual activemq::commands::BaseCommand::~~BaseCommand ()`
[inline, virtual]

6.109.2 Member Function Documentation

6.109.2.1 `virtual void activemq::commands::BaseCommand::copyDataStructure (const DataStructure * src)` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 173), `activemq::commands::ActiveMQBytesMessage` (p. 201), `activemq::commands::ActiveMQMapMessage` (p. 312), `activemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQObjectMessage` (p. 384), `activemq::commands::ActiveMQStreamMessage` (p. 467), `activemq::commands::ActiveMQTextMessage` (p. 575), `activemq::commands::BrokerError` (p. 746), `activemq::commands::BrokerInfo` (p. 777), `activemq::commands::ConnectionControl` (p. 1136), `activemq::commands::ConnectionError` (p. 1161), `activemq::commands::ConnectionInfo` (p. 1213), `activemq::commands::ConsumerControl` (p. 1251), `activemq::commands::ConsumerInfo` (p. 1302), `activemq::commands::ControlCommand` (p. 1331), `activemq::commands::DataArrayResponse` (p. 1357), `activemq::commands::DataResponse` (p. 1396), `activemq::commands::DestinationInfo` (p. 1485), `activemq::commands::ExceptionResponse` (p. 1583), `activemq::commands::FlushCommand` (p. 1677), `activemq::commands::IntegerResponse` (p. 1778), `activemq::commands::KeepAliveInfo` (p. 1931), `activemq::commands::Message` (p. 2150), `activemq::commands::MessageAck` (p. 2189), `activemq::commands::MessageDispatch` (p. 2220), `activemq::commands::MessageDispatchNotification` (p. 2252), `activemq::commands::MessagePull` (p. 2347), `activemq::commands::ProducerAck` (p. 2580), `activemq::commands::ProducerInfo` (p. 2632), `activemq::commands::RemoveInfo` (p. 2704), `activemq::commands::RemoveSubscriptionInfo` (p. 2728), `activemq::commands::ReplayCommand` (p. 2753), `activemq::commands::Response` (p. 2782), `activemq::commands::SessionInfo` (p. 2878), `activemq::commands::ShutdownInfo` (p. 2935), `activemq::commands::TransactionInfo` (p. 3241), and `activemq::commands::WireFormatInfo` (p. 3372).

References `getCommandId()`, and `isResponseRequired()`.

6.109.2.2 `virtual bool activemq::commands::BaseCommand::equals (const DataStructure * value) const` [inline, virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 174), `activemq::commands::ActiveMQBytesMessage` (p. 201), `activemq::commands::ActiveMQMapMessage` (p. 312), `activemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQMessageTemplate< T >` (p. 369), `activemq::commands::ActiveMQObjectMessage` (p. 384), `activemq::commands::ActiveMQStreamMessage` (p. 468), `activemq::commands::ActiveMQTextMessage` (p. 575), `activemq::commands::BrokerInfo` (p. 777), `activemq::commands::ConnectionControl` (p. 1137), `activemq::commands::ConnectionError` (p. 1161), `activemq::commands::ConnectionInfo` (p. 1213), `activemq::commands::ConsumerControl` (p. 1252), `activemq::commands::ConsumerInfo` (p. 1303), `activemq::commands::ControlCommand` (p. 1331), `activemq::commands::DataArrayResponse` (p. 1357), `activemq::commands::DataResponse` (p. 1396), `activemq::commands::DestinationInfo` (p. 1486), `activemq::commands::ExceptionResponse` (p. 1583), `activemq::commands::FlushCommand` (p. 1677), `activemq::commands::IntegerResponse` (p. 1778), `activemq::commands::KeepAliveInfo` (p. 1931), `activemq::commands::Message` (p. 2150), `activemq::commands::MessageAck` (p. 2190), `activemq::commands::MessageDispatch` (p. 2221), `activemq::commands::MessageDispatchNotification` (p. 2253), `activemq::commands::MessagePull` (p. 2348), `activemq::commands::ProducerAck` (p. 2580), `activemq::commands::ProducerInfo` (p. 2633), `activemq::commands::RemoveInfo` (p. 2704), `activemq::commands::RemoveSubscriptionInfo` (p. 2729), `activemq::commands::ReplayCommand` (p. 2753), `activemq::commands::Response` (p. 2782), `activemq::commands::SessionInfo` (p. 2878), `activemq::commands::ShutdownInfo` (p. 2935), `activemq::commands::TransactionInfo` (p. 3241), `activemq::commands::WireFormatInfo` (p. 3372), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 369), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 369).

References `activemq::commands::BaseDataStructure::equals()`.

6.109.2.3 `virtual int activemq::commands::BaseCommand::getCommandId () const` [inline, virtual]

Gets the **Command** (p. 1063) Id of this **Message** (p. 2145).

Returns:

Command (p. 1063) Id

Implements `activemq::commands::Command` (p. 1064).

Referenced by `copyDataStructure()`.

6.109.2.4 virtual bool activemq::commands::BaseCommand::isBrokerInfo () const
[inline, virtual]

Implements **activemq::commands::Command** (p. 1064).

Reimplemented in **activemq::commands::BrokerInfo** (p. 779).

6.109.2.5 virtual bool activemq::commands::BaseCommand::isConnectionInfo ()
const [inline, virtual]

Implements **activemq::commands::Command** (p. 1064).

Reimplemented in **activemq::commands::ConnectionInfo** (p. 1214).

6.109.2.6 virtual bool activemq::commands::BaseCommand::isConsumerInfo ()
const [inline, virtual]

Implements **activemq::commands::Command** (p. 1064).

Reimplemented in **activemq::commands::ConsumerInfo** (p. 1304).

6.109.2.7 virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ()
const [inline, virtual]

Implements **activemq::commands::Command** (p. 1064).

Reimplemented in **activemq::commands::KeepAliveInfo** (p. 1932).

6.109.2.8 virtual bool activemq::commands::BaseCommand::isMessage () const
[inline, virtual]

Implements **activemq::commands::Command** (p. 1064).

Reimplemented in **activemq::commands::Message** (p. 2156).

6.109.2.9 virtual bool activemq::commands::BaseCommand::isMessageAck () const
[inline, virtual]

Implements **activemq::commands::Command** (p. 1065).

Reimplemented in **activemq::commands::MessageAck** (p. 2191).

6.109.2.10 virtual bool activemq::commands::BaseCommand::isMessageDispatch ()
const [inline, virtual]

Implements **activemq::commands::Command** (p. 1065).

Reimplemented in **activemq::commands::MessageDispatch** (p. 2222).

6.109.2.11 `virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1065).

Reimplemented in `activemq::commands::MessageDispatchNotification` (p. 2254).

6.109.2.12 `virtual bool activemq::commands::BaseCommand::isProducerAck () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1065).

Reimplemented in `activemq::commands::ProducerAck` (p. 2581).

6.109.2.13 `virtual bool activemq::commands::BaseCommand::isProducerInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1065).

Reimplemented in `activemq::commands::ProducerInfo` (p. 2634).

6.109.2.14 `virtual bool activemq::commands::BaseCommand::isRemoveInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1065).

Reimplemented in `activemq::commands::RemoveInfo` (p. 2705).

6.109.2.15 `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1065).

Reimplemented in `activemq::commands::RemoveSubscriptionInfo` (p. 2730).

6.109.2.16 `virtual bool activemq::commands::BaseCommand::isResponse () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 1065).

Reimplemented in `activemq::commands::Response` (p. 2783).

6.109.2.17 `virtual bool activemq::commands::BaseCommand::isResponseRequired () const [inline, virtual]`

Is a **Response** (p. 2781) required for this **Command** (p. 1063).

Returns:

true if a response is required.

Implements **activemq::commands::Command** (p. 1066).

Referenced by `copyDataStructure()`.

6.109.2.18 `virtual bool activemq::commands::BaseCommand::isShutdownInfo ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1066).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 2936).

6.109.2.19 `virtual bool activemq::commands::BaseCommand::isTransactionInfo ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1066).

Reimplemented in **activemq::commands::TransactionInfo** (p. 3242).

6.109.2.20 `virtual bool activemq::commands::BaseCommand::isWireFormatInfo ()`
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1066).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 3376).

6.109.2.21 `virtual void activemq::commands::BaseCommand::setCommandId (int`
`id) [inline, virtual]`

Sets the **Command** (p. 1063) Id of this **Message** (p. 2145).

Parameters:

id **Command** (p. 1063) Id

Implements **activemq::commands::Command** (p. 1066).

6.109.2.22 `virtual void activemq::commands::BaseCommand::setResponseRequired`
`(const bool required) [inline, virtual]`

Set if this **Message** (p. 2145) requires a **Response** (p. 2781).

Parameters:

required true if response is required

Implements **activemq::commands::Command** (p. 1066).

6.109.2.23 `virtual std::string activemq::commands::BaseCommand::toString ()`
`const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Implements **activemq::commands::Command** (p. 1067).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 176), **activemq::commands::ActiveMQBytesMessage** (p. 208), **activemq::commands::ActiveMQMapMessage** (p. 320), **activemq::commands::ActiveMQMessage** (p. 344), **activemq::commands::ActiveMQObjectMessage** (p. 385), **activemq::commands::ActiveMQStreamMessage** (p. 474), **activemq::commands::ActiveMQTextMessage** (p. 577), **activemq::commands::BrokerInfo** (p. 780), **activemq::commands::ConnectionControl** (p. 1138), **activemq::commands::ConnectionError** (p. 1162), **activemq::commands::ConnectionInfo** (p. 1215), **activemq::commands::ConsumerControl** (p. 1253), **activemq::commands::ConsumerInfo** (p. 1305), **activemq::commands::ControlCommand** (p. 1332), **activemq::commands::DataArrayResponse** (p. 1358), **activemq::commands::DataResponse** (p. 1397), **activemq::commands::DestinationInfo** (p. 1487), **activemq::commands::ExceptionResponse** (p. 1584), **activemq::commands::FlushCommand** (p. 1678), **activemq::commands::IntegerResponse** (p. 1779), **activemq::commands::KeepAliveInfo** (p. 1932), **activemq::commands::Message** (p. 2159), **activemq::commands::MessageAck** (p. 2192), **activemq::commands::MessageDispatch** (p. 2222), **activemq::commands::MessageDispatchNotification** (p. 2255), **activemq::commands::MessagePull** (p. 2349), **activemq::commands::ProducerAck** (p. 2581), **activemq::commands::ProducerInfo** (p. 2634), **activemq::commands::RemoveInfo** (p. 2705), **activemq::commands::RemoveSubscriptionInfo** (p. 2730), **activemq::commands::ReplayCommand** (p. 2754), **activemq::commands::Response** (p. 2783), **activemq::commands::SessionInfo** (p. 2879), **activemq::commands::ShutdownInfo** (p. 2936), **activemq::commands::TransactionInfo** (p. 3243), and **activemq::commands::WireFormatInfo** (p. 3378).

References **activemq::commands::BaseDataStructure::toString()**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

6.110 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 657).

#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.110.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 657). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.110.2 Constructor & Destructor Documentation

6.110.2.1 `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

6.110.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

6.110.3 Member Function Documentation

6.110.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 178), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 323), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 387), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 480), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 579), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 784), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1141), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1165), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1218), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1256), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1310), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1335), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1360), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1594), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1688), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1789), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 1942), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2203), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2240), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`

(p. 2266), [activemq::wireformat::openwire::marshal::v3::MessageMarshaller](#) (p. 2316), [activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller](#) (p. 2356), [activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller](#) (p. 2588), [activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller](#) (p. 2653), [activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller](#) (p. 2716), [activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller](#) (p. 2745), [activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller](#) (p. 2761), [activemq::wireformat::openwire::marshal::v3::ResponseMarshaller](#) (p. 2797), [activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller](#) (p. 2898), [activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller](#) (p. 2946), and [activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller](#) (p. 3254).

6.110.3.2 virtual void [activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal](#) ([OpenWireFormat](#) * *wireFormat*, [commands::DataStructure](#) * *dataStructure*, [decaf::io::DataInputStream](#) * *dataIn*) throw ([decaf::io::IOException](#)) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - [BinaryReader](#) that provides that data source

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1436).

Reimplemented in [activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller](#) (p. 179), [activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller](#) (p. 215), [activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller](#) (p. 324), [activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller](#) (p. 347), [activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller](#) (p. 388), [activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller](#) (p. 481), [activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller](#) (p. 580), [activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller](#) (p. 785), [activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller](#) (p. 1142), [activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller](#) (p. 1166), [activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller](#) (p. 1219), [activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller](#) (p. 1257), [activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller](#) (p. 1311), [activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller](#) (p. 1336), [activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller](#) (p. 1361), [activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller](#) (p. 1400), [activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller](#) (p. 1495), [activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller](#) (p. 1595), [activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller](#) (p. 1689), [activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller](#) (p. 1790), [activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller](#) (p. 1943),

activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2204), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2241), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 2267), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2316), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2357), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2589), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2654), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2717), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2746), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2762), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2798), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2899), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2947), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3255).

6.110.3.3 virtual int activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1
(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 179), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 215), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 324), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 481), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 580), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 785), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1166), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1219), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1257), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1311), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1336), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1361),

activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1400),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1595),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1790),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1943),
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2204),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2241),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2267), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2317),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2357),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2654),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2717),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2746), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2762), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2798),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2899),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2947), and
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3255).

6.110.3.4 virtual void `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2`
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`
 (p. 179), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`
 (p. 215), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`
 (p. 324), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`
 (p. 347), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`
 (p. 388), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`
 (p. 481), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`
 (p. 580), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 785),
`activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1142),
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1166),
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1219),

activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1257),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1311),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1336),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1362),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1401),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1596),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1791),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1943),
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2204),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2241),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2267), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2318),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2357),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2654),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2717),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2746), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2762), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2799),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2899),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2947), and
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3255).

6.110.3.5 virtual void ac-
 tivemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`
 (p. 180), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`
 (p. 216), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`
 (p. 325), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`
 (p. 348), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`
 (p. 389), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`
 (p. 482), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`

(p. 581), [activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller](#) (p. 786), [activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller](#) (p. 1143), [activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller](#) (p. 1167), [activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller](#) (p. 1220), [activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller](#) (p. 1258), [activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller](#) (p. 1312), [activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller](#) (p. 1337), [activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller](#) (p. 1362), [activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller](#) (p. 1401), [activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller](#) (p. 1496), [activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller](#) (p. 1596), [activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller](#) (p. 1690), [activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller](#) (p. 1791), [activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller](#) (p. 1944), [activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller](#) (p. 2205), [activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller](#) (p. 2242), [activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller](#) (p. 2268), [activemq::wireformat::openwire::marshal::v3::MessageMarshaller](#) (p. 2318), [activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller](#) (p. 2358), [activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller](#) (p. 2590), [activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller](#) (p. 2655), [activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller](#) (p. 2718), [activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller](#) (p. 2747), [activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller](#) (p. 2763), [activemq::wireformat::openwire::marshal::v3::ResponseMarshaller](#) (p. 2800), [activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller](#) (p. 2900), [activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller](#) (p. 2948), and [activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller](#) (p. 3256).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h`

6.111 activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 664).

#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.111.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 664). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.111.2 Constructor & Destructor Documentation

- 6.111.2.1 `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMarshaller()` `[inline]`
- 6.111.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~~BaseCommandMarshaller()` `[inline, virtual]`

6.111.3 Member Function Documentation

- 6.111.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions:

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 182), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 218), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 327), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 391), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 484), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 583), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 788), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1145), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1169), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1226), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1314), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1343), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1368), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1684), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1785), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1950), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2211), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2244), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`

(p. 2270), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2326), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2360), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2600), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2649), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 2708), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 2741), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 2773), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 2882), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 2938), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3246).

6.111.3.2 virtual void `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal`
**(`OpenWireFormat * wireFormat`, `commands::DataStructure`
`* dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`)** [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 183), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 328), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 485), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 584), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 789), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1146), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1170), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1227), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1265), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1344), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1369), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1404), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1685), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1786), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1951),

activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2212), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2245), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2271), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2326), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2361), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2601), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2650), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2709), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2742), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2774), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2808), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2883), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2939), and activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3247).

6.111.3.3 virtual int activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 219), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 328), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 485), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 584), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 789), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1146), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1170), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1227), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1265), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1315), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1344), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1369),

activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1404),
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1503), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1591),
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1685),
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1786),
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1951),
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2212), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2245), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2271), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2327),
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2361),
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2601),
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2650),
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2709), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 2742), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 2774), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2808),
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2883), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 2939), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3247).

6.111.3.4 virtual void ac-

activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 219), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 328), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 485), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 584), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 789), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1146), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1170), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1227),

activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1265),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1315),
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1344),
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1370),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1405),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1503),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1592),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1685),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1787),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1951),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2212),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2245),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2271), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2328),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2361),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2601),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2650),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2709),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2742), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2774), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2809),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2883),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2939), and
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3247).

6.111.3.5 virtual void `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal`
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`
 (p. 184), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`
 (p. 220), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`
 (p. 329), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`
 (p. 352), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`
 (p. 393), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`
 (p. 486), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`

(p. 585), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 790), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1147), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1171), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1228), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1266), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1345), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1370), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1405), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1504), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1592), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1686), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1787), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1952), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2213), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2246), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2272), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2328), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2362), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2602), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2651), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 2710), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 2743), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 2775), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2810), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 2884), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 2940), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3248).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h`

6.112 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 671).

#include <src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.112.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 671). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.112.2 Constructor & Destructor Documentation

6.112.2.1 `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

6.112.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

6.112.3 Member Function Documentation

6.112.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 488), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 587), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 792), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1149), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1173), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1222), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1260), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1318), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1339), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1364), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1598), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1692), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1793), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1946), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2207), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2236), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`

(p. 2262), [activemq::wireformat::openwire::marshal::v4::MessageMarshaller](#) (p. 2311), [activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller](#) (p. 2368), [activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller](#) (p. 2592), [activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller](#) (p. 2645), [activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller](#) (p. 2720), [activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller](#) (p. 2749), [activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller](#) (p. 2769), [activemq::wireformat::openwire::marshal::v4::ResponseMarshaller](#) (p. 2812), [activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller](#) (p. 2886), [activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller](#) (p. 2942), and [activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller](#) (p. 3250).

6.112.3.2 virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1436).

Reimplemented in [activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller](#) (p. 187), [activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller](#) (p. 223), [activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller](#) (p. 332), [activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller](#) (p. 355), [activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller](#) (p. 396), [activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller](#) (p. 489), [activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller](#) (p. 588), [activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller](#) (p. 793), [activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller](#) (p. 1150), [activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller](#) (p. 1174), [activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller](#) (p. 1223), [activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller](#) (p. 1261), [activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller](#) (p. 1319), [activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller](#) (p. 1340), [activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller](#) (p. 1365), [activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller](#) (p. 1412), [activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller](#) (p. 1499), [activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller](#) (p. 1599), [activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller](#) (p. 1693), [activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller](#) (p. 1794), [activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller](#) (p. 1947),

activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2208), **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller** (p. 2237), **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller** (p. 2263), **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2311), **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** (p. 2369), **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller** (p. 2593), **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller** (p. 2646), **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller** (p. 2721), **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller** (p. 2750), **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller** (p. 2770), **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2813), **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller** (p. 2887), **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller** (p. 2943), and **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller** (p. 3251).

6.112.3.3 virtual int activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal1
(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 332), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 355), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 396), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 489), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 588), **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 793), **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** (p. 1150), **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller** (p. 1174), **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller** (p. 1223), **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller** (p. 1261), **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller** (p. 1319), **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller** (p. 1340), **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1365),

activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1499),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1599),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1693),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1794),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1947),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2208),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2237),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2263), activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2312),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2369),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2593),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2646),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2721),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 2750), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 2770), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2813),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2887),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2943), and
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3251).

6.112.3.4 virtual void `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal2`
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`
 (p. 187), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`
 (p. 223), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
 (p. 332), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
 (p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
 (p. 396), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 489), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 588), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 793),
`activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1150),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1174),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1223),

activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1261),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1319),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1340),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1366),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1413),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1499),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1600),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1693),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1795),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1947),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2208),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2237),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2263), activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2313),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2369),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2593),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2646),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2721),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 2750), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 2770), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2814),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2887),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2943), and
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3251).

6.112.3.5 virtual void ac-
 tivemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`
 (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`
 (p. 224), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
 (p. 333), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
 (p. 356), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
 (p. 397), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 490), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`

(p. 589), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 794), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1151), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1175), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1224), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1262), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1320), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1341), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1366), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1413), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1500), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1600), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1694), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1795), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1948), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2209), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2238), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2264), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2370), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2594), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2647), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2722), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 2751), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 2771), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2815), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2888), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2944), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3252).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h`

6.113 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 678).

#include <src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.113.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 678). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.113.2 Constructor & Destructor Documentation

- 6.113.2.1** `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::BaseCommandMarshaller()` `[inline]`
- 6.113.2.2** `virtual activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::~~BaseCommandMarshaller()` `[inline, virtual]`

6.113.3 Member Function Documentation

- 6.113.3.1** `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions:

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 190), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 226), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 399), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 591), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 796), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1153), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1177), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1230), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1268), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1322), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1347), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1506), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1602), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1696), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1797), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1938), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2199), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2248), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`

(p. 2274), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2364), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2596), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2641), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2724), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 2733), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 2765), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2894), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2954), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3258).

6.113.3.2 virtual void `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseUnmarshal`
**(`OpenWireFormat * wireFormat`, `commands::DataStructure`
`* dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`)** [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 191), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 336), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 359), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 493), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 592), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 797), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1154), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1178), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1231), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1269), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1323), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1348), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1373), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1507), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1603), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1697), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1798), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1939),

activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2200),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2249),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (p. 2275),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2321),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2365),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2597),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2642),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2725),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (p. 2734),
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 2766),
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2803),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2895),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2955), and
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3259).

6.113.3.3 virtual int `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal1`
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 191),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 227),
`activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 336),
`activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 359),
`activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 400),
`activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 493),
`activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 592),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 797),
`activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1154),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1178),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1231),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1269),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1323),
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1348),
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1373),

activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1408),
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1507), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1603),
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1697),
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1798),
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1939),
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2200), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2249), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2275),
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2322),
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2365),
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2597),
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2642),
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2725), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 2734),
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 2766), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2803),
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2895), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 2955), and
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3259).

6.113.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal2
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 191), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 336), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 359), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 400), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 493), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 592), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 797), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1154), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1178), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1231),

activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1269),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1323),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1348),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1374),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1409),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1507),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1604),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1697),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1799),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1939),
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2200),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2249),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2275), activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2323),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2365),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2597),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2642),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2725),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2734), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2766), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2804),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2895),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2955), and
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3259).

6.113.3.5 virtual void `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightUnmarshal`
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 192), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 228), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 337), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 360), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
 (p. 401), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 494), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`

(p. 593), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 798), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1155), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1179), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1232), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1270), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1349), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1374), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1409), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1508), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1604), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1698), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1799), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1940), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2201), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2250), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2276), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2366), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2598), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2643), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2726), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 2735), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 2767), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2805), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2896), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2956), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3260).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h`

6.114 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 685).

#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.114.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 685). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.114.2 Constructor & Destructor Documentation

6.114.2.1 `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMarshaller()` [inline]

6.114.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~~BaseCommandMarshaller()` [inline, virtual]

6.114.3 Member Function Documentation

6.114.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 194), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 339), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 496), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 595), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 800), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1157), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1181), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1234), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1272), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1326), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1351), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1490), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1680), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1781), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 1934), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2195), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2232), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`

(p. 2258), [activemq::wireformat::openwire::marshal::v2::MessageMarshaller](#) (p. 2306), [activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller](#) (p. 2352), [activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller](#) (p. 2584), [activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller](#) (p. 2637), [activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller](#) (p. 2712), [activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller](#) (p. 2737), [activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller](#) (p. 2757), [activemq::wireformat::openwire::marshal::v2::ResponseMarshaller](#) (p. 2792), [activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller](#) (p. 2890), [activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller](#) (p. 2950), and [activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller](#) (p. 3262).

6.114.3.2 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1436).

Reimplemented in [activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller](#) (p. 195), [activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller](#) (p. 231), [activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller](#) (p. 340), [activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller](#) (p. 363), [activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller](#) (p. 404), [activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller](#) (p. 497), [activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller](#) (p. 596), [activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller](#) (p. 801), [activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller](#) (p. 1158), [activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller](#) (p. 1182), [activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller](#) (p. 1235), [activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller](#) (p. 1273), [activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller](#) (p. 1327), [activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller](#) (p. 1352), [activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller](#) (p. 1377), [activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller](#) (p. 1416), [activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller](#) (p. 1491), [activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller](#) (p. 1587), [activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller](#) (p. 1681), [activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller](#) (p. 1782), [activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller](#) (p. 1935),

activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2196), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2233), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 2259), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2306), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 2353), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2585), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2638), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2713), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2738), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2758), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2793), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2891), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2951), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 3263).

6.114.3.3 virtual int activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1
(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 195), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 231), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 340), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 363), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 497), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 596), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 801), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1158), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1182), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1235), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1273), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1327), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1352), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1377),

activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1416),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1491),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1587),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1782),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1935),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2196),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2233),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2259), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2307),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2353),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2585),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2638),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2713),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2738), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2758), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2793),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2891),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2951), and
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3263).

6.114.3.4 virtual void `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2`
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`
 (p. 195), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`
 (p. 231), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`
 (p. 340), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`
 (p. 363), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`
 (p. 404), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`
 (p. 497), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`
 (p. 596), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 801),
`activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1158),
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1182),
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1235),

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1327),
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1352),
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1378),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1417),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1491),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1588),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1783),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1935),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2196),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2233),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2259), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2308),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2353),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2585),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2638),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2713),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2738), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2758), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2794),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2891),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2951), and
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3263).

6.114.3.5 virtual void ac-
 tivemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`
 (p. 196), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`
 (p. 232), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`
 (p. 341), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`
 (p. 364), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`
 (p. 405), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`
 (p. 498), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`

(p. 597), [activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller](#) (p. 802), [activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller](#) (p. 1159), [activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller](#) (p. 1183), [activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller](#) (p. 1236), [activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller](#) (p. 1274), [activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller](#) (p. 1328), [activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller](#) (p. 1353), [activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller](#) (p. 1378), [activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller](#) (p. 1417), [activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller](#) (p. 1492), [activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller](#) (p. 1588), [activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller](#) (p. 1682), [activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller](#) (p. 1783), [activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller](#) (p. 1936), [activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller](#) (p. 2197), [activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller](#) (p. 2234), [activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller](#) (p. 2260), [activemq::wireformat::openwire::marshal::v2::MessageMarshaller](#) (p. 2308), [activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller](#) (p. 2354), [activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller](#) (p. 2586), [activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller](#) (p. 2639), [activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller](#) (p. 2714), [activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller](#) (p. 2739), [activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller](#) (p. 2759), [activemq::wireformat::openwire::marshal::v2::ResponseMarshaller](#) (p. 2795), [activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller](#) (p. 2892), [activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller](#) (p. 2952), and [activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller](#) (p. 3264).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h`

6.115 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Inherits `activemq::wireformat::openwire::marshal::DataStreamMarshaller`.

Inherited by `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`,
`activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`,
`activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller`,
`activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller`,
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller`,
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::SessionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`,
`activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`,
`activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller`,
`activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller`,
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller`,
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`,
`activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`,
`activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller`, `ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller`, `ac-`

tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller,	and ac-
tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller.	

Public Member Functions

- virtual `~BaseDataStreamMarshaller()`
- virtual `int tightMarshal(OpenWireFormat *format, AMQCPP_UNUSED, commands::DataStructure *command, AMQCPP_UNUSED, utils::BooleanStream *bs, AMQCPP_UNUSED) throw (decaf::io::IOException)`

Tight Marshal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *format **AMQCPP_UNUSED**, **commands::DataStructure** *command **AMQCPP_UNUSED**, **decaf::io::DataOutputStream** *ds **AMQCPP_UNUSED**, **utils::BooleanStream** *bs **AMQCPP_UNUSED**) throw (**decaf::io::IOException**)

Tight Marshal to the given stream.

- virtual void **tightUnmarshal** (**OpenWireFormat** *format **AMQCPP_UNUSED**, **commands::DataStructure** *command **AMQCPP_UNUSED**, **decaf::io::DataInputStream** *dis **AMQCPP_UNUSED**, **utils::BooleanStream** *bs **AMQCPP_UNUSED**) throw (**decaf::io::IOException**)

Tight Un-Marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *format **AMQCPP_UNUSED**, **commands::DataStructure** *command **AMQCPP_UNUSED**, **decaf::io::DataOutputStream** *ds **AMQCPP_UNUSED**) throw (**decaf::io::IOException**)

Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *format **AMQCPP_UNUSED**, **commands::DataStructure** *command **AMQCPP_UNUSED**, **decaf::io::DataInputStream** *dis **AMQCPP_UNUSED**) throw (**decaf::io::IOException**)

Loose Un-Marshal to the given stream.

Static Public Member Functions

- static std::string **toString** (const **commands::MessageId** *id)
Converts the object to a String.
- static std::string **toString** (const **commands::ProducerId** *id)
Converts the object to a String.
- static std::string **toString** (const **commands::TransactionId** *txnId)
Converts the given transaction ID into a String.
- static std::string **toHexFromBytes** (const std::vector< unsigned char > &data)
given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Protected Member Functions

- virtual **commands::DataStructure** * **tightUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tight Unmarshal the cached object.

- virtual int **tightMarshalCachedObject1** (OpenWireFormat *wireFormat, commands::DataStructure *data, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.
- virtual void **tightMarshalCachedObject2** (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tightly marshals the passed DataStructure based object to the passed streams returning nothing.
- virtual void **looseMarshalCachedObject** (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Loosely marshals the passed DataStructure based object to the passed stream returning nothing.
- virtual commands::DataStructure * **looseUnmarshalCachedObject** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Loose Unmarshal the cached object.
- virtual int **tightMarshalNestedObject1** (OpenWireFormat *wireFormat, commands::DataStructure *object, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.
- virtual void **tightMarshalNestedObject2** (OpenWireFormat *wireFormat, commands::DataStructure *object, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tightly marshals the passed DataStructure based object to the passed streams returning nothing.
- virtual commands::DataStructure * **tightUnmarshalNestedObject** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tight Unmarshal the nested object.
- virtual commands::DataStructure * **looseUnmarshalNestedObject** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Loose Unmarshal the nested object.
- virtual void **looseMarshalNestedObject** (OpenWireFormat *wireFormat, commands::DataStructure *object, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Loose marshall the nested object.
- virtual std::string **tightUnmarshalString** (decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Performs Tight Unmarshaling of String Objects.

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tight Marshals the passed string to the streams passed.

- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Loose Marshal the String to the DataOuputStream passed.

- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Loose Un-Marshall the String to the DataOuputStream passed.

- virtual int **tightMarshalLong1** (**OpenWireFormat** *wireFormat, long long value, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tightly marshal the long long to the BooleanStream passed.

- virtual void **tightMarshalLong2** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tightly marshal the long long to the Streams passed.

- virtual long long **tightUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tight marshal the long long type.

- virtual void **looseMarshalLong** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Tightly marshal the long long to the BooleanStream passed.

- virtual long long **looseUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Loose marshal the long long type.

- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Tight Unmarshal an array of char.

- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Loose Unmarshal an array of char.

- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs, int size) throw (**decaf::io::IOException**)

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual `std::vector< unsigned char > looseUnmarshalConstByteArray (decaf::io::DataInputStream *dataIn, int size) throw (decaf::io::IOException)`

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual `commands::DataStructure * tightUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tight Unmarshal the Error object.

- virtual `int tightMarshalBrokerError1 (OpenWireFormat *wireFormat, commands::DataStructure *data, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tight Marshal the Error object.

- virtual `void tightMarshalBrokerError2 (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tight Marshal the Error object.

- virtual `commands::DataStructure * looseUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Loose Unmarshal the Error object.

- virtual `void looseMarshalBrokerError (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Tight Marshal the Error object.

- `template<typename T > int tightMarshalObjectArray1 (OpenWireFormat *wireFormat, std::vector< T > objects, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

- `template<typename T > void tightMarshalObjectArray2 (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- `template<typename T > void looseMarshalObjectArray (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- virtual std::string readAsciiString (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

6.115.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

Since:

2.0

6.115.2 Constructor & Destructor Documentation

- 6.115.2.1 virtual
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller()
[inline, virtual]

6.115.3 Member Function Documentation

- 6.115.3.1 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataOutputStream *ds *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

ds - DataOutputStream to marshal to

Exceptions:

IOException if an error occurs.

- 6.115.3.2 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

Exceptions:

IOException if an error occurs.

6.115.3.3 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedObject(OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException)
[protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

Parameters:

wireFormat - The OpenWireFormat properties

data - DataStructure Object Pointer to marshal

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.115.3.4 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong(OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException)
[protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters:

wireFormat - The OpenWireFormat properties

value - long long to marshal

dataOut - DataOutputStream to marshal to.

Exceptions:

IOException if an error occurs.

6.115.3.5 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject(OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException)
[protected, virtual]

Loose marshal the nested object.

Parameters:

wireFormat - The OpenwireFormat properties
object - DataStructure Object Pointer to marshal
dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.115.3.6 `template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectA
 (OpenWireFormat * wireFormat, std::vector< T > objects,
 decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)
 [inline, protected]`

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters:

wireFormat - The OpenwireFormat properties
objects - array of DataStructure object pointers.
dataOut - stream to write marshalled data to

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

6.115.3.7 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString
 (const std::string value, decaf::io::DataOutputStream * dataOut) throw (
 decaf::io::IOException) [protected, virtual]`

Loose Marshal the String to the DataOutputStream passed.

Parameters:

value - string to marshal
dataOut - stream to write marshaled form to

Exceptions:

IOException if an error occurs.

6.115.3.8 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataInputStream *dis *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Loose Un-Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from

Exceptions:

IOException if an error occurs.

6.115.3.9 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBroken (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshalled form from

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.115.3.10 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBytes (decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal an array of char.

Parameters:

dataIn - the DataInputStream to Un-Marshal from

Returns:

the unmarshalled vector of chars.

Exceptions:

IOException if an error occurs.

6.115.3.11 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCached(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Unmarshal the cached object.

Parameters:

wireFormat - The OpenWireFormat properties

dataIn - stream to read marshaled form from

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.115.3.12 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConstrained(decaf::io::DataInputStream * dataIn, int size) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters:

dataIn - the DataInputStream to Un-Marshall from

size - size of the const array to unmarshal

Returns:

the unmarshaled vector of chars.

Exceptions:

IOException if an error occurs.

6.115.3.13 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose marshal the long long type.

Parameters:

wireFormat - The OpenWireFormat properties

dataIn - stream to read marshaled form from

Returns:

the unmarshaled long long

Exceptions:

IOException if an error occurs.

6.115.3.14 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNested(OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal the nested object.

Parameters:

wireFormat - The OpenWireFormat properties

dataIn - stream to read marshaled form from

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.115.3.15 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString(decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Un-Marshal the String to the DataOutputStream passed.

Parameters:

dataIn - stream to read marshaled form from

Returns:

the unmarshaled string

Exceptions:

IOException if an error occurs.

6.115.3.16 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString(decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

Parameters:

dataIn - DataInputStream to read from

Returns:

string value read from stream

6.115.3.17 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenWireFormat properties

command - the object to Marshal

bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

6.115.3.18 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataOutputStream *ds *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenWireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

6.115.3.19 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

data - Error to Marshal

bs - boolean stream to marshal to.

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

6.115.3.20 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

6.115.3.21 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached(OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters:

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

Returns:

size of data written.

Exceptions:

IOException if an error occurs.

6.115.3.22 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached
(OpenWireFormat * *wireFormat*, commands::DataStructure * *data*,
decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*)
throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters:

wireFormat - The OpenWireFormat properties

data - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.115.3.23 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1
(OpenWireFormat * *wireFormat*, long long *value*, utils::BooleanStream
* *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters:

wireFormat - The OpenWireFormat properties

value - long long to marshal

bs - boolean stream to marshal to.

Returns:

size of data written.

Exceptions:

IOException if an error occurs.

6.115.3.24 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2 (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshal the long long to the Streams passed.

Parameters:

wireFormat - The OpenWireFormat properties

value - long long to marshal

dataOut - stream to write marshaled form to

bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

6.115.3.25 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters:

wireFormat - The OpenWireFormat properties

object - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

Returns:

size of data written.

Exceptions:

IOException if an error occurs.

6.115.3.26 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters:

wireFormat - The OpenWireFormat properties

object - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

```
6.115.3.27  template<typename T > int ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject
             (OpenWireFormat * wireFormat, std::vector< T > objects,
             utils::BooleanStream * bs) throw ( decaf::io::IOException ) [inline,
             protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

Parameters:

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

bs - boolean stream to marshal to.

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

```
6.115.3.28  template<typename T > void ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject
             (OpenWireFormat * wireFormat, std::vector< T > objects,
             decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs)
             throw ( decaf::io::IOException ) [inline, protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters:

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

dataOut - stream to write marshalled data to

bs - boolean stream to marshal to.

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

6.115.3.29 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 (const std::string & *value*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

Parameters:

value - string to marshal
bs - BooleanStream to use.

Returns:

size of marshaled string.

Exceptions:

IOException if an error occurs.

6.115.3.30 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 (const std::string & *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshals the passed string to the streams passed.

Parameters:

value - string to marshal
dataOut - the DataOutputStream to Marshal to
bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

6.115.3.31 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal (OpenWireFormat **format* AMQCPP_UNUSED, commands::DataStructure **command* AMQCPP_UNUSED, decaf::io::DataInputStream **dis* AMQCPP_UNUSED, utils::BooleanStream **bs* AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]

Tight Un-Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties
command - the object to Un-Marshall
dis - the DataInputStream to Un-Marshall from
bs - boolean stream to Un-Marshall from.

Exceptions:

IOException if an error occurs.

6.115.3.32 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBroken(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshall the Error object.

Parameters:

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshalled form from
bs - boolean stream to marshal to.

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.115.3.33 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByteArray(decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal an array of char.

Parameters:

dataIn - the DataInputStream to Un-Marshall from
bs - boolean stream to unmarshal from.

Returns:

the unmarshaled vector of chars.

Exceptions:

IOException if an error occurs.

6.115.3.34 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCached(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal the cached object.

Parameters:

wireFormat - The OpenWireFormat properties
dataIn - stream to read marshaled form from
bs - boolean stream to marshal to.

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.115.3.35 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConsistent(decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters:

dataIn - the DataInputStream to Un-Marshall from
bs - boolean stream to unmarshal from.
size - size of the const array to unmarshal

Returns:

the unmarshaled vector of chars.

Exceptions:

IOException if an error occurs.

6.115.3.36 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight marshal the long long type.

Parameters:

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshaled form from
bs - boolean stream to marshal to.

Returns:

the unmarshaled long long

Exceptions:

IOException if an error occurs.

6.115.3.37 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNested(OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [protected, virtual]`

Tight Unmarshal the nested object.

Parameters:

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshaled form from
bs - boolean stream to marshal to.

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.115.3.38 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString(decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [protected, virtual]`

Performs Tight Unmarshaling of String Objects.

Parameters:

dataIn - the DataInputStream to Un-Marshal from
bs - boolean stream to unmarshal from.

Returns:

the unmarshaled string.

Exceptions:

IOException if an error occurs.

6.115.3.39 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes (const std::vector< unsigned char > & *data*) [static]

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Parameters:

data - unsigned char data array pointer

Returns:

a string coded in hex that represents the data

6.115.3.40 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::TransactionId * *txnId*) [static]

Converts the given transaction ID into a String.

Parameters:

txnId - TransactionId poitner

Returns:

string representation of the id

6.115.3.41 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::ProducerId * *id*) [static]

Converts the object to a String.

Parameters:

id - ProducerId pointer

Returns:

string representing the id

6.115.3.42 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::MessageId * *id*) [static]

Converts the object to a String.

Parameters:

id - MessageId pointer

Returns:

string representing the id

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h`

6.116 activemq::commands::BaseDataStructure Class Reference

#include <src/main/activemq/commands/BaseDataStructure.h> Inheritance diagram for activemq::commands::BaseDataStructure:

Public Member Functions

- virtual `~BaseDataStructure ()`
- virtual bool `isMarshalAware ()` const
Determine if this object is aware of marshaling and should have its before and after marshaling methods called.
- virtual void `beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed before an marshal.
- virtual void `afterMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed after an unmarshal.
- virtual void `beforeUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed before an unmarshal.
- virtual void `afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Perform any processing needed after an unmarshal.
- virtual void `setMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)`
Called to set the data to this object that will contain the objects marshaled form.
- virtual std::vector< unsigned char > `getMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED)`
Called to get the data to this object that will contain the objects marshaled form.
- virtual void `copyDataStructure (const DataStructure *src AMQCPP_UNUSED)`
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string `toString ()` const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value AMQCPP_UNUSED)` const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*

6.116.1 Constructor & Destructor Documentation

6.116.1.1 `virtual activemq::commands::BaseDataStructure::~~BaseDataStructure ()`
[inline, virtual]

6.116.2 Member Function Documentation

6.116.2.1 `virtual void activemq::commands::BaseDataStructure::afterMarshal`
(`wireformat::WireFormat *wireFormat AMQCPP_UNUSED`) throw (`decaf::io::IOException`) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters:

wireformat - the OpenWireFormat object in use.

6.116.2.2 `virtual void activemq::commands::BaseDataStructure::afterUnmarshal`
(`wireformat::WireFormat *wireFormat AMQCPP_UNUSED`) throw (`decaf::io::IOException`) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters:

wireformat - the OpenWireFormat object in use.

Reimplemented in `activemq::commands::Message` (p. 2149), and `activemq::commands::WireFormatInfo` (p. 3371).

6.116.2.3 `virtual void activemq::commands::BaseDataStructure::beforeMarshal`
(`wireformat::WireFormat *wireFormat AMQCPP_UNUSED`) throw (`decaf::io::IOException`) [inline, virtual]

Perform any processing needed before an marshal.

Parameters:

wireformat - the OpenWireFormat object in use.

Reimplemented in `activemq::commands::Message` (p. 2149), and `activemq::commands::WireFormatInfo` (p. 3372).

6.116.2.4 `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal`
(`wireformat::WireFormat *wireFormat AMQCPP_UNUSED`) throw (`decaf::io::IOException`) [inline, virtual]

Perform any processing needed before an unmarshal.

Parameters:

wireformat - the OpenWireFormat object in use.

6.116.2.5 virtual void activemq::commands::BaseDataStructure::copyDataStructure (const DataStructure *src *AMQCPP_UNUSED*) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented in `activemq::commands::BooleanExpression` (p. 738).

6.116.2.6 virtual bool activemq::commands::BaseDataStructure::equals (const DataStructure *value *AMQCPP_UNUSED*) const [inline, virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

6.116.2.7 virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) [inline, virtual]

Called to get the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the wireformat object to control unmarshaling

Returns:

buffer that holds the objects data.

6.116.2.8 virtual bool activemq::commands::BaseDataStructure::isMarshalAware () const [inline, virtual]

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

Returns:

true if aware of marshaling

Implements `activemq::wireformat::MarshalAware` (p. 2120).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 316), `activemq::commands::Message` (p. 2155), and `activemq::commands::WireFormatInfo` (p. 3374).

6.116.2.9 `virtual void activemq::commands::BaseDataStructure::setMarshaledForm (wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)` [inline, virtual]

Called to set the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the wireformat object to control unmarshaling

data - vector of object binary data

6.116.2.10 `virtual std::string activemq::commands::BaseDataStructure::toString () const` [inline, virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p.1465).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p.176), **activemq::commands::ActiveMQBytesMessage** (p.208), **activemq::commands::ActiveMQDestination** (p.282), **activemq::commands::ActiveMQMapMessage** (p.320), **activemq::commands::ActiveMQMessage** (p.344), **activemq::commands::ActiveMQObjectMessage** (p.385), **activemq::commands::ActiveMQQueue** (p.422), **activemq::commands::ActiveMQStreamMessage** (p.474), **activemq::commands::ActiveMQTempDestination** (p.501), **activemq::commands::ActiveMQTempQueue** (p.526), **activemq::commands::ActiveMQTempTopic** (p.551), **activemq::commands::ActiveMQTextMessage** (p.577), **activemq::commands::ActiveMQTopic** (p.601), **activemq::commands::BaseCommand** (p.655), **activemq::commands::BooleanExpression** (p.738), **activemq::commands::BrokerId** (p.753), **activemq::commands::BrokerInfo** (p.780), **activemq::commands::Command** (p.1067), **activemq::commands::ConnectionControl** (p.1138), **activemq::commands::ConnectionError** (p.1162), **activemq::commands::ConnectionId** (p.1189), **activemq::commands::ConnectionInfo** (p.1215), **activemq::commands::ConsumerControl** (p.1253), **activemq::commands::ConsumerId** (p.1278), **activemq::commands::ConsumerInfo** (p.1305), **activemq::commands::ControlCommand** (p.1332), **activemq::commands::DataArrayResponse** (p.1358), **activemq::commands::DataResponse** (p.1397), **activemq::commands::DestinationInfo** (p.1487), **activemq::commands::DiscoveryEvent** (p.1513), **activemq::commands::ExceptionResponse** (p.1584), **activemq::commands::FlushCommand** (p.1678), **activemq::commands::IntegerResponse** (p.1779), **activemq::commands::JournalQueueAck** (p.1836), **activemq::commands::JournalTopicAck** (p.1861), **activemq::commands::JournalTrace** (p.1885), **activemq::commands::JournalTransaction** (p.1908), **activemq::commands::KeepAliveInfo** (p.1932), **activemq::commands::LastPartialCommand** (p.1960), **activemq::commands::LocalTransactionId** (p.1996), **activemq::commands::Message** (p.2159), **activemq::commands::MessageAck**

(p. 2192), [activemq::commands::MessageDispatch](#) (p. 2222), [activemq::commands::MessageDispatchNotification](#) (p. 2255), [activemq::commands::MessageId](#) (p. 2282), [activemq::commands::MessagePull](#) (p. 2349), [activemq::commands::NetworkBridgeFilter](#) (p. 2395), [activemq::commands::PartialCommand](#) (p. 2475), [activemq::commands::ProducerAck](#) (p. 2581), [activemq::commands::ProducerId](#) (p. 2609), [activemq::commands::ProducerInfo](#) (p. 2634), [activemq::commands::RemoveInfo](#) (p. 2705), [activemq::commands::RemoveSubscriptionInfo](#) (p. 2730), [activemq::commands::ReplayCommand](#) (p. 2754), [activemq::commands::Response](#) (p. 2783), [activemq::commands::SessionId](#) (p. 2856), [activemq::commands::SessionInfo](#) (p. 2879), [activemq::commands::ShutdownInfo](#) (p. 2936), [activemq::commands::SubscriptionInfo](#) (p. 3100), [activemq::commands::TransactionId](#) (p. 3219), [activemq::commands::TransactionInfo](#) (p. 3243), [activemq::commands::WireFormatInfo](#) (p. 3378), and [activemq::commands::XATransactionId](#) (p. 3413).

Referenced by [activemq::commands::BooleanExpression::toString\(\)](#), and [activemq::commands::BaseCommand::toString\(\)](#).

The documentation for this class was generated from the following file:

- [src/main/activemq/commands/BaseDataStructure.h](#)

6.117 `binary_function` Class Reference

Inheritance diagram for `binary_function`:

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.118 decaf::net::BindException Class Reference

#include <src/main/decaf/net/BindException.h> Inheritance diagram for decaf::net::BindException:

Public Member Functions

- **BindException** () throw ()
Default Constructor.
- **BindException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **BindException** (const **BindException** &ex) throw ()
Copy Constructor.
- **BindException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BindException** (const std::exception *cause) throw ()
Constructor.
- **BindException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BindException** * clone () const
Clones this exception.
- virtual ~**BindException** () throw ()

6.118.1 Constructor & Destructor Documentation

6.118.1.1 decaf::net::BindException::BindException () throw () [inline]

Default Constructor.

6.118.1.2 decaf::net::BindException::BindException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.118.1.3 `decaf::net::BindException::BindException (const BindException & ex)
throw () [inline]`

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.118.1.4 `decaf::net::BindException::BindException (const char * file, const int
lineNumber, const std::exception * cause, const char * msg, ...) throw
() [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.118.1.5 `decaf::net::BindException::BindException (const std::exception * cause)
throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.118.1.6 `decaf::net::BindException::BindException (const char * file, const int
lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.118.1.7 `virtual decaf::net::BindException::~~BindException () throw ()` [inline, virtual]

6.118.2 Member Function Documentation

6.118.2.1 `virtual BindException* decaf::net::BindException::clone () const`
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2973).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BindException.h`

6.119 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

#include <src/main/decaf/io/BlockingByteArrayInputStream.h> Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

Public Member Functions

- **BlockingByteArrayInputStream** ()
Default Constructor - uses a default internal buffer.
- **BlockingByteArrayInputStream** (const unsigned char *buffer, std::size_t bufferSize)
Constructor that initializes the internal buffer.
- virtual ~**BlockingByteArrayInputStream** ()
Destructor.
- virtual void **setByteArray** (const unsigned char *buffer, std::size_t bufferSize)
Sets the data that this reader uses.
- virtual std::size_t **available** () const throw (IOException)
Indicates the number of bytes available to be read without blocking.
- virtual int **read** () throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (io::IOException)
Closes the target input stream.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.119.1 Detailed Description

This is a blocking version of a byte buffer stream. Read operations block until the requested data becomes available in the internal buffer via a call to setByteArray.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ()

Default Constructor - uses a default internal buffer.

6.119.2.2 `decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char * buffer, std::size_t bufferSize)`

Constructor that initializes the internal buffer.

See also:

`setByteArray` (p. 729).

6.119.2.3 `virtual decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream ()` [virtual]

Destructor.

6.119.3 Member Function Documentation

6.119.3.1 `virtual std::size_t decaf::io::BlockingByteArrayInputStream::available () const throw (IOException)` [inline, virtual]

Indicates the number of bytes available to be read without blocking.

Returns:

the data available in the internal buffer.

Exceptions:

IOException (p. 1820) if an error occurs.

Implements `decaf::io::InputStream` (p. 1741).

6.119.3.2 `virtual void decaf::io::BlockingByteArrayInputStream::close () throw (io::IOException)` [virtual]

Closes the target input stream.

Exceptions:

IOException (p. 1820) if an error occurs.

Implements `decaf::io::Closeable` (p. 1019).

6.119.3.3 `virtual void decaf::io::BlockingByteArrayInputStream::lock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.119.3.4 virtual void decaf::io::BlockingByteArrayInputStream::mark (int readLimit *DECAF_UNUSED*) [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1741).

6.119.3.5 virtual bool decaf::io::BlockingByteArrayInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1741).

6.119.3.6 virtual void decaf::io::BlockingByteArrayInputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3124).

6.119.3.7 virtual void decaf::io::BlockingByteArrayInputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

6.119.3.8 `virtual int decaf::io::BlockingByteArrayInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException) [virtual]`

Reads an array of bytes from the buffer. If the desired amount of data is not currently available, this operation will block until the appropriate amount of data is available in the buffer via a call to `setByteArray`.

Parameters:

buffer (out) the target buffer
offset the position in the buffer to start from.
bufferSize the size of the output buffer.

Returns:

the number of bytes read. or -1 if EOF

Exceptions:

IOException (p. 1820) if an error occurs.

Implements **decaf::io::InputStream** (p. 1742).

6.119.3.9 `virtual int decaf::io::BlockingByteArrayInputStream::read () throw (IOException) [virtual]`

Reads a single byte from the buffer. This operation will block until data has been added to the buffer via a call to `setByteArray`.

Returns:

the next byte.

Exceptions:

IOException (p. 1820) if an error occurs.

Implements **decaf::io::InputStream** (p. 1742).

6.119.3.10 `virtual void decaf::io::BlockingByteArrayInputStream::reset () throw (IOException) [inline, virtual]`

Repositions this stream to the position at the time the `mark` method was last called on this input stream. If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 1820) might be thrown. * If such an **IOException** (p. 1820) is not thrown, then the stream is reset

to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1820). * If an **IOException** (p. 1820) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException (p. 1820)

Implements **decaf::io::InputStream** (p. 1742).

**6.119.3.11 virtual void decaf::io::BlockingByteArrayInputStream::setByteArray
(const unsigned char * *buffer*, std::size_t *bufferSize*) [virtual]**

Sets the data that this reader uses. Replaces any existing data and resets the read index to the beginning of the buffer. When this method is called, it notifies any other threads that data is now available to be read.

Parameters:

buffer The new data to be copied to the internal buffer.

bufferSize The size of the new buffer.

**6.119.3.12 virtual std::size_t decaf::io::BlockingByteArrayInputStream::skip
(std::size_t *num*) throw (io::IOException,
lang::exceptions::UnsupportedOperationException) [virtual]**

Skips over and discards *n* bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If *n* is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1740) creates a byte array and then repeatedly reads into it until *n* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException (p. 1820) if an error occurs

Implements **decaf::io::InputStream** (p. 1743).

6.119.3.13 `virtual bool decaf::io::BlockingByteArrayInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3126).

6.119.3.14 `virtual void decaf::io::BlockingByteArrayInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3127).

6.119.3.15 `virtual void decaf::io::BlockingByteArrayInputStream::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.119.3.16 virtual void decaf::io::BlockingByteArrayInputStream::wait (long long *milliseconds*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.119.3.17 virtual void decaf::io::BlockingByteArrayInputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BlockingByteArrayInputStream.h**

6.120 decaf::lang::Boolean Class Reference

#include <src/main/decaf/lang/Boolean.h> Inheritance diagram for decaf::lang::Boolean:

Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual ~**Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const
*Compares this **Boolean** (p. 732) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Boolean** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const
*Compares this **Boolean** (p. 732) instance with another.*
- virtual bool **operator==** (const bool &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const bool &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const bool &b) const

Static Public Member Functions

- static **Boolean** **valueOf** (bool value)
- static **Boolean** **valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)
Parses the String passed and extracts an bool.
- static std::string **toString** (bool value)
Converts the bool to a String representation.

Static Public Attributes

- static const **Boolean** **_FALSE**

The Class object representing the primitive false boolean.

- static const **Boolean** **_TRUE**

The Class object representing the primitive type boolean.

6.120.1 Constructor & Destructor Documentation

6.120.1.1 decaf::lang::Boolean::Boolean (bool *value*)

Parameters:

value - primitive boolean to wrap.

6.120.1.2 decaf::lang::Boolean::Boolean (const std::string & *value*)

Parameters:

value - String value to convert to a boolean.

6.120.1.3 virtual decaf::lang::Boolean::~~Boolean () [inline, virtual]

6.120.2 Member Function Documentation

6.120.2.1 bool decaf::lang::Boolean::booleanValue () const [inline]

Returns:

the primitive boolean value of this object

6.120.2.2 virtual int decaf::lang::Boolean::compareTo (const bool & *b*) const [virtual]

Compares this **Boolean** (p. 732) instance with another.

Parameters:

b - the **Boolean** (p. 732) instance to be compared

Returns:

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **bool** > (p. 1083).

6.120.2.3 `virtual int decaf::lang::Boolean::compareTo (const Boolean & b) const`
[virtual]

Compares this **Boolean** (p. 732) instance with another.

Parameters:

b - the **Boolean** (p. 732) instance to be compared

Returns:

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1083).

6.120.2.4 `bool decaf::lang::Boolean::equals (const bool & b) const` [inline,
virtual]

Returns:

true if the two **Boolean** (p. 732) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 1084).

6.120.2.5 `bool decaf::lang::Boolean::equals (const Boolean & b) const` [inline,
virtual]

Returns:

true if the two **Boolean** (p. 732) Objects have the same value.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1084).

6.120.2.6 `virtual bool decaf::lang::Boolean::operator< (const bool & value) const`
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1084).

6.120.2.7 virtual bool decaf::lang::Boolean::operator< (const Boolean & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Boolean >** (p. 1084).

6.120.2.8 virtual bool decaf::lang::Boolean::operator== (const bool & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< bool >** (p. 1085).

6.120.2.9 virtual bool decaf::lang::Boolean::operator== (const Boolean & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Boolean >** (p. 1085).

6.120.2.10 static bool decaf::lang::Boolean::parseBoolean (const std::string & *value*) [static]

Parses the String passed and extracts an bool.

Parameters:

value The std::string value to parse

Returns:

bool value

6.120.2.11 `static std::string decaf::lang::Boolean::toString (bool value) [static]`

Converts the bool to a String representation.

Parameters:

value The bool value to convert.

Returns:

std::string representation of the bool value passed.

6.120.2.12 `std::string decaf::lang::Boolean::toString () const`**Returns:**

the string representation of this Booleans value.

6.120.2.13 `static Boolean decaf::lang::Boolean::valueOf (const std::string & value) [static]`**Parameters:**

value The std::string value to convert to a Boolean (p. 732) instance.

Returns:

a **Boolean** (p. 732) instance of the string value

6.120.2.14 `static Boolean decaf::lang::Boolean::valueOf (bool value) [static]`**Parameters:**

value The bool value to convert to a Boolean (p. 732) instance.

Returns:

a **Boolean** (p. 732) instance of the primitive boolean value

6.120.3 **Field Documentation****6.120.3.1** `const Boolean decaf::lang::Boolean::_FALSE [static]`

The Class object representing the primitive false boolean.

6.120.3.2 `const Boolean decaf::lang::Boolean::_TRUE [static]`

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

6.121 activemq::commands::BooleanExpression Class Reference

#include <src/main/activemq/commands/BooleanExpression.h> Inheritance diagram for activemq::commands::BooleanExpression:

Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** ()
- virtual **DataStructure * cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*

6.121.1 Constructor & Destructor Documentation

- 6.121.1.1 **activemq::commands::BooleanExpression::BooleanExpression** ()
[inline]
- 6.121.1.2 **virtual activemq::commands::BooleanExpression::~~BooleanExpression** ()
[inline, virtual]

6.121.2 Member Function Documentation

- 6.121.2.1 **virtual DataStructure* activemq::commands::BooleanExpression::cloneDataStructure** () const
[inline, virtual]

Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.121.2.2 `virtual void activemq::commands::BooleanExpression::copyDataStructure (const DataStructure *src AMQCPP_UNUSED) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::BaseDataStructure` (p. 717).

6.121.2.3 `virtual bool activemq::commands::BooleanExpression::equals (const DataStructure * value) const [inline, virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

References `activemq::commands::BaseDataStructure::equals()`.

6.121.2.4 `virtual std::string activemq::commands::BooleanExpression::toString () const [inline, virtual]`

Returns a string containing the information for this `DataStructure` (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 718).

References `activemq::commands::BaseDataStructure::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BooleanExpression.h`

6.122 activemq::wireformat::openwire::utils::BooleanStream Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>
```

Public Member Functions

- **BooleanStream** ()
- virtual **~BooleanStream** ()
- bool **readBoolean** () throw (decaf::io::IOException)
Read a boolean data element from the internal data buffer.
- void **writeBoolean** (bool value) throw (decaf::io::IOException)
Writes a Boolean value to the internal data buffer.
- void **marshal** (decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Marshal the data to a DataOutputStream.
- void **marshal** (std::vector< unsigned char > &dataOut)
Marshal the data to a STL vector of unsigned chars.
- void **unmarshal** (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Unmarshal a Boolean data stream from the Input Stream.
- void **clear** ()
Clears to old position markers, data starts at the beginning.
- int **marshalledSize** ()
Calc the size that data is marshalled to.

6.122.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`. The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 `activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ()`

6.122.2.2 `virtual
activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream
() [inline, virtual]`

6.122.3 Member Function Documentation

6.122.3.1 `void activemq::wireformat::openwire::utils::BooleanStream::clear ()`

Clears to old position markers, data starts at the beginning.

6.122.3.2 `void activemq::wireformat::openwire::utils::BooleanStream::marshal
(std::vector< unsigned char > & dataOut)`

Marshal the data to a STL vector of unsigned chars.

Parameters:

dataOut - reference to a vector to write the data to.

6.122.3.3 `void activemq::wireformat::openwire::utils::BooleanStream::marshal
(decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)`

Marshal the data to a DataOutputStream.

Parameters:

dataOut - Stream to write the data to.

6.122.3.4 `int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize
()`

Calc the size that data is marshalled to.

Returns:

int size of marshalled data.

6.122.3.5 `bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean
() throw (decaf::io::IOException)`

Read a boolean data element from the internal data buffer.

Returns:

boolean from the stream

6.122.3.6 void activemq::wireformat::openwire::utils::BooleanStream::unmarshal
(decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException)

Unmarshal a Boolean data stream from the Input Stream.

Parameters:

dataIn - Input Stream to read data from.

6.122.3.7 void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean
(bool *value*) throw (decaf::io::IOException)

Writes a Boolean value to the internal data buffer.

Parameters:

value - boolean data to write.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**BooleanStream.h**

6.123 decaf::util::concurrent::BrokenBarrierException Class Reference

#include <src/main/decaf/util/concurrent/BrokenBarrierException.h> Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

Public Member Functions

- **BrokenBarrierException** () throw ()
Default Constructor.
- **BrokenBarrierException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **BrokenBarrierException** (const BrokenBarrierException &ex) throw ()
Copy Constructor.
- **BrokenBarrierException** (const std::exception *cause) throw ()
Constructor.
- **BrokenBarrierException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BrokenBarrierException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BrokenBarrierException** * clone () const
Clones this exception.
- virtual ~**BrokenBarrierException** () throw ()

6.123.1 Constructor & Destructor Documentation

6.123.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException () throw () [inline]

Default Constructor.

6.123.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.123.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const BrokenBarrierException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.123.1.4 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.123.1.5 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
msg - The message to report
... - list of primitives that are formatted into the message

6.123.1.6 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
cause - The exception that was the cause for this one to be thrown.
msg - The message to report
... - list of primitives that are formatted into the message

6.123.1.7 **virtual**
 decaf::util::concurrent::BrokenBarrierException::~~BrokenBarrierException
 () throw () [inline, virtual]

6.123.2 Member Function Documentation

6.123.2.1 **virtual BrokenBarrierException* de-**
 caf::util::concurrent::BrokenBarrierException::clone ()
 const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

 a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BrokenBarrierException.h`

6.124 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

#include <src/main/activemq/commands/BrokerError.h> Inheritance diagram for activemq::commands::BrokerError:

Data Structures

- struct **StackTraceElement**

Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1461) Type as defined in **CommandTypes.h**.*
- virtual **BrokerError** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual const std::string & **getMessage** () const
Gets the string holding the error message.
- virtual void **setMessage** (const std::string &message)
*Sets the string that contains the error **Message** (p. 2145).*
- virtual const std::string & **getExceptionClass** () const
Gets the string holding the Exception Class name.
- virtual void **setExceptionClass** (const std::string &exceptionClass)
Sets the string that contains the Exception Class name.
- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const
Gets the Broker Error that caused this exception.
- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)
Sets the Broker Error that caused this exception.

- virtual const std::vector< **decaf::lang::Pointer< StackTraceElement > > & getStackTraceElements** () const

Gets the Stack Trace Elements for the Exception.

- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer< StackTraceElement > > & stackTraceElements**)

Sets the Stack Trace Elements for this Exception.

6.124.1 Detailed Description

This class represents an Exception sent from the Broker. The Broker sends java Throwables, so we must mimic its structure here.

6.124.2 Constructor & Destructor Documentation

6.124.2.1 **activemq::commands::BrokerError::BrokerError** () [inline]

6.124.2.2 **virtual activemq::commands::BrokerError::~~BrokerError** () [virtual]

6.124.3 Member Function Documentation

6.124.3.1 **virtual BrokerError* activemq::commands::BrokerError::cloneDataStructure** ()
const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

References **copyDataStructure()**.

6.124.3.2 **virtual void activemq::commands::BrokerError::copyDataStructure**
(const **DataStructure * src**) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

Referenced by **cloneDataStructure()**.

6.124.3.3 `virtual const decaf::lang::Pointer<BrokerError>&
activemq::commands::BrokerError::getCause () const [inline, virtual]`

Gets the Broker Error that caused this exception.

Returns:

Broker Error Pointer

6.124.3.4 `virtual unsigned char ac-
tivismq::commands::BrokerError::getDataStructureType ()
const [inline, virtual]`

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1464).

6.124.3.5 `virtual const std::string& ac-
tivismq::commands::BrokerError::getExceptionClass ()
const [inline, virtual]`

Gets the string holding the Exception Class name.

Returns:

Exception Class name

6.124.3.6 `virtual const std::string& activemq::commands::BrokerError::getMessage
() const [inline, virtual]`

Gets the string holding the error message.

Returns:

String Message (p. 2145)

6.124.3.7 `virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >&
activemq::commands::BrokerError::getStackTraceElements () const
[inline, virtual]`

Gets the Stack Trace Elements for the Exception.

Returns:

Stack Trace Elements

6.124.3.8 `virtual void activemq::commands::BrokerError::setCause (const decaf::lang::Pointer< BrokerError > & cause)` [inline, virtual]

Sets the Broker Error that caused this exception.

Parameters:

cause - Broker Error

6.124.3.9 `virtual void activemq::commands::BrokerError::setExceptionClass (const std::string & exceptionClass)` [inline, virtual]

Sets the string that contains the Exception Class name.

Parameters:

exceptionClass - String Exception Class name

6.124.3.10 `virtual void activemq::commands::BrokerError::setMessage (const std::string & message)` [inline, virtual]

Sets the string that contains the error **Message** (p.2145).

Parameters:

message - String Error **Message** (p. 2145)

6.124.3.11 `virtual void activemq::commands::BrokerError::setStackTraceElements (const std::vector< decaf::lang::Pointer< StackTraceElement > > & stackTraceElements)` [inline, virtual]

Sets the Stack Trace Elements for this Exception.

Parameters:

stackTraceElements - Stack Trace Elements

6.124.3.12 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1067).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.125 activemq::exceptions::BrokerException Class Reference

#include <src/main/activemq/exceptions/BrokerException.h> Inheritance diagram for activemq::exceptions::BrokerException:

Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const exceptions::ActiveMQException &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
- **BrokerException** (const char *file, const int lineNumber, const commands::BrokerError *error) throw ()
- virtual **BrokerException** * **clone** () const
Clones this exception.
- virtual ~**BrokerException** () throw ()

6.125.1 Constructor & Destructor Documentation

- 6.125.1.1 **activemq::exceptions::BrokerException::BrokerException** () throw ()
[inline]
- 6.125.1.2 **activemq::exceptions::BrokerException::BrokerException** (const exceptions::ActiveMQException & *ex*) throw () [inline]
- 6.125.1.3 **activemq::exceptions::BrokerException::BrokerException** (const **BrokerException** & *ex*) throw () [inline]
- 6.125.1.4 **activemq::exceptions::BrokerException::BrokerException** (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]
- 6.125.1.5 **activemq::exceptions::BrokerException::BrokerException** (const char * *file*, const int *lineNumber*, const commands::BrokerError * *error*) throw () [inline]

References decaf::lang::Exception::setMark(), and decaf::lang::Exception::setMessage().

- 6.125.1.6 **virtual activemq::exceptions::BrokerException::~~BrokerException** ()
throw () [inline, virtual]

6.125.2 Member Function Documentation

- 6.125.2.1 **virtual BrokerException*** **activemq::exceptions::BrokerException::clone** () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **activemq::exceptions::ActiveMQException** (p. 306).

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/BrokerException.h`

6.126 activemq::commands::BrokerId Class Reference

#include <src/main/activemq/commands/BrokerId.h> Inheritance diagram for activemq::commands::BrokerId:

Public Types

- typedef decaf::lang::PointerComparator< BrokerId > COMPARATOR

Public Member Functions

- **BrokerId** ()
- **BrokerId** (const **BrokerId** &other)
- virtual ~**BrokerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **BrokerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **BrokerId** &value) const
- virtual bool **equals** (const **BrokerId** &value) const
- virtual bool **operator==** (const **BrokerId** &value) const
- virtual bool **operator<** (const **BrokerId** &value) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

Static Public Attributes

- static const unsigned char **ID_BROKERID** = 124

Protected Attributes

- std::string value

6.126.1 Member Typedef Documentation

6.126.1.1 `typedef decaf::lang::PointerComparator<BrokerId>
activemq::commands::BrokerId::COMPARATOR`

6.126.2 Constructor & Destructor Documentation

6.126.2.1 `activemq::commands::BrokerId::BrokerId ()`

6.126.2.2 `activemq::commands::BrokerId::BrokerId (const BrokerId & other)`

6.126.2.3 `virtual activemq::commands::BrokerId::~~BrokerId () [virtual]`

6.126.3 Member Function Documentation

6.126.3.1 `virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.126.3.2 `virtual int activemq::commands::BrokerId::compareTo (const BrokerId &
value) const [virtual]`

6.126.3.3 `virtual void activemq::commands::BrokerId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

6.126.3.4 `virtual bool activemq::commands::BrokerId::equals (const BrokerId &
value) const [virtual]`

6.126.3.5 `virtual bool activemq::commands::BrokerId::equals (const DataStructure
* value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.126.3.6 virtual unsigned char activemq::commands::BrokerId::getDataStructureType ()
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.126.3.7 virtual std::string& activemq::commands::BrokerId::getValue ()
[virtual]

6.126.3.8 virtual const std::string& activemq::commands::BrokerId::getValue ()
const [virtual]

6.126.3.9 virtual bool activemq::commands::BrokerId::operator< (const BrokerId
& *value*) const [virtual]

6.126.3.10 BrokerId& activemq::commands::BrokerId::operator= (const BrokerId
& *other*)

6.126.3.11 virtual bool activemq::commands::BrokerId::operator== (const
BrokerId & *value*) const [virtual]

6.126.3.12 virtual void activemq::commands::BrokerId::setValue (const std::string
& *value*) [virtual]

6.126.3.13 virtual std::string activemq::commands::BrokerId::toString () const
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 718).

6.126.4 Field Documentation

6.126.4.1 const unsigned char activemq::commands::BrokerId::ID_BROKERID =
124 [static]

6.126.4.2 std::string activemq::commands::BrokerId::value [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/`**BrokerId.h**

6.127 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 755).

#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.127.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 755). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.127.2 Constructor & Destructor Documentation

6.127.2.1 `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

6.127.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

6.127.3 Member Function Documentation

6.127.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.127.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.127.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.127.3.4 virtual void **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal** (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.127.3.5 virtual int **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1** (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.127.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.127.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h**

6.128 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 759).

#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.128.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 759). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.128.2 Constructor & Destructor Documentation

6.128.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

6.128.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

6.128.3 Member Function Documentation

6.128.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.128.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.128.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.128.3.4 virtual void **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.128.3.5 virtual int **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.128.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.128.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerIdMarshaller.h**

6.129 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 763).

#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.129.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 763). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.129.2 Constructor & Destructor Documentation

6.129.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

6.129.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

6.129.3 Member Function Documentation

6.129.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.129.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.129.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.129.3.4 virtual void **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.129.3.5 virtual int **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.129.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.129.3.7 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h**

6.130 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 767).

#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.130.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 767). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.130.2 Constructor & Destructor Documentation

6.130.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

6.130.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

6.130.3 Member Function Documentation

6.130.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.130.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.130.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.130.3.4 virtual void **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.130.3.5 virtual int **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.130.3.6 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.130.3.7 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerIdMarshaller.h**

6.131 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 771).

#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.131.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 771). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.131.2 Constructor & Destructor Documentation

6.131.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller()` [inline]

6.131.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~~BrokerIdMarshaller()` [inline, virtual]

6.131.3 Member Function Documentation

6.131.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.131.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.131.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.131.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.131.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.131.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.131.3.7 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h**

6.132 activemq::commands::BrokerInfo Class Reference

#include <src/main/activemq/commands/BrokerInfo.h> Inheritance diagram for activemq::commands::BrokerInfo:

Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **BrokerInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool masterBroker)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool faultTolerantConfiguration)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool duplexConnection)

- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool **networkConnection**)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long **connectionId**)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &**brokerUploadUrl**)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &**networkProperties**)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_BROKERINFO** = 2

Protected Member Functions

- **BrokerInfo** (const **BrokerInfo** &)
- **BrokerInfo** & **operator=** (const **BrokerInfo** &)

Protected Attributes

- **Pointer< BrokerId > brokerId**
- std::string **brokerURL**
- std::vector< decaf::lang::Pointer< **BrokerInfo** > > **peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

6.132.1 Constructor & Destructor Documentation

6.132.1.1 `activemq::commands::BrokerInfo::BrokerInfo (const BrokerInfo &)`
[inline, protected]

6.132.1.2 `activemq::commands::BrokerInfo::BrokerInfo ()`

6.132.1.3 `virtual activemq::commands::BrokerInfo::~~BrokerInfo ()` [virtual]

6.132.2 Member Function Documentation

6.132.2.1 `virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.132.2.2 `virtual void activemq::commands::BrokerInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.132.2.3 `virtual bool activemq::commands::BrokerInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

- 6.132.2.4 `virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ()`
[virtual]
- 6.132.2.5 `virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () const`
[virtual]
- 6.132.2.6 `virtual std::string& activemq::commands::BrokerInfo::getBrokerName ()`
[virtual]
- 6.132.2.7 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerName () const`
[virtual]
- 6.132.2.8 `virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ()`
[virtual]
- 6.132.2.9 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl () const` [virtual]
- 6.132.2.10 `virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ()`
[virtual]
- 6.132.2.11 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL () const`
[virtual]
- 6.132.2.12 `virtual long long activemq::commands::BrokerInfo::getConnectionId () const` [virtual]
- 6.132.2.13 `virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.132.2.14 `virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties ()`
[virtual]
- 6.132.2.15 `virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties ()`
`const` [virtual]
- 6.132.2.16 `virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ()` [virtual]
- 6.132.2.17 `virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ()` `const`
[virtual]
- 6.132.2.18 `virtual bool activemq::commands::BrokerInfo::isBrokerInfo ()` `const`
[inline, virtual]

Returns:

an answer of true to the `isBrokerInfo()` (p. 779) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 653).

- 6.132.2.19 virtual bool activemq::commands::BrokerInfo::isDuplexConnection () const [virtual]
- 6.132.2.20 virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration () const [virtual]
- 6.132.2.21 virtual bool activemq::commands::BrokerInfo::isMasterBroker () const [virtual]
- 6.132.2.22 virtual bool activemq::commands::BrokerInfo::isNetworkConnection () const [virtual]
- 6.132.2.23 virtual bool activemq::commands::BrokerInfo::isSlaveBroker () const [virtual]
- 6.132.2.24 BrokerInfo& activemq::commands::BrokerInfo::operator= (const BrokerInfo &) [inline, protected]
- 6.132.2.25 virtual void activemq::commands::BrokerInfo::setBrokerId (const Pointer< BrokerId > & *brokerId*) [virtual]
- 6.132.2.26 virtual void activemq::commands::BrokerInfo::setBrokerName (const std::string & *brokerName*) [virtual]
- 6.132.2.27 virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const std::string & *brokerUploadUrl*) [virtual]
- 6.132.2.28 virtual void activemq::commands::BrokerInfo::setBrokerURL (const std::string & *brokerURL*) [virtual]
- 6.132.2.29 virtual void activemq::commands::BrokerInfo::setConnectionId (long long *connectionId*) [virtual]
- 6.132.2.30 virtual void activemq::commands::BrokerInfo::setDuplexConnection (bool *duplexConnection*) [virtual]
- 6.132.2.31 virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration (bool *faultTolerantConfiguration*) [virtual]
- 6.132.2.32 virtual void activemq::commands::BrokerInfo::setMasterBroker (bool *masterBroker*) [virtual]
- 6.132.2.33 virtual void activemq::commands::BrokerInfo::setNetworkConnection (bool *networkConnection*) [virtual]
- 6.132.2.34 virtual void activemq::commands::BrokerInfo::setNetworkProperties (const std::string & *networkProperties*) [virtual]
- 6.132.2.35 virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos (const std::vector< decaf::lang::Pointer< BrokerInfo > > & *peerBrokerInfos*) [virtual]
- 6.132.2.36 virtual void activemq::commands::BrokerInfo::setSlaveBroker (bool *slaveBroker*) [virtual]
- 6.132.2.37 virtual std::string activemq::commands::BrokerInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.132.2.38 `virtual Pointer<Command> activemq::commands::BrokerInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.132.3 Field Documentation

- 6.132.3.1 `Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId` [protected]
- 6.132.3.2 `std::string activemq::commands::BrokerInfo::brokerName` [protected]
- 6.132.3.3 `std::string activemq::commands::BrokerInfo::brokerUploadUrl` [protected]
- 6.132.3.4 `std::string activemq::commands::BrokerInfo::brokerURL` [protected]
- 6.132.3.5 `long long activemq::commands::BrokerInfo::connectionId` [protected]
- 6.132.3.6 `bool activemq::commands::BrokerInfo::duplexConnection` [protected]
- 6.132.3.7 `bool activemq::commands::BrokerInfo::faultTolerantConfiguration` [protected]
- 6.132.3.8 `const unsigned char activemq::commands::BrokerInfo::ID_ -
BROKERINFO = 2` [static]
- 6.132.3.9 `bool activemq::commands::BrokerInfo::masterBroker` [protected]
- 6.132.3.10 `bool activemq::commands::BrokerInfo::networkConnection` [protected]
- 6.132.3.11 `std::string activemq::commands::BrokerInfo::networkProperties` [protected]
- 6.132.3.12 `std::vector< decaf::lang::Pointer<BrokerInfo> >
activemq::commands::BrokerInfo::peerBrokerInfos` [protected]
- 6.132.3.13 `bool activemq::commands::BrokerInfo::slaveBroker` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerInfo.h`

6.133 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 783).

#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.133.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 783). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.133.2 Constructor & Destructor Documentation

6.133.2.1 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.133.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.133.3 Member Function Documentation

6.133.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.133.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.133.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.133.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.133.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.133.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.133.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h**

6.134 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 787).

#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.134.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 787). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.134.2 Constructor & Destructor Documentation

6.134.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.134.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.134.3 Member Function Documentation

6.134.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.134.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.134.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.134.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.134.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.134.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.134.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerInfoMarshaller.h**

6.135 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 791).

#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.135.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 791). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.135.2 Constructor & Destructor Documentation

6.135.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.135.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.135.3 Member Function Documentation

6.135.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.135.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.135.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

```
6.135.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

```
6.135.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal1
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.135.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.135.3.7 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerInfoMarshaller.h**

6.136 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 795).

#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.136.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 795). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.136.2 Constructor & Destructor Documentation

6.136.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.136.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.136.3 Member Function Documentation

6.136.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.136.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.136.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.136.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.136.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.136.3.6 virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.136.3.7 virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerInfoMarshaller.h**

6.137 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 799).

#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.137.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 799). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.137.2 Constructor & Destructor Documentation

6.137.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.137.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.137.3 Member Function Documentation

6.137.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.137.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.137.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.137.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.137.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.137.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.137.3.7 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h**

6.138 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

#include <src/main/decaf/nio/Buffer.h> Inheritance diagram for decaf::nio::Buffer:

Public Member Functions

- **Buffer** (std::size_t capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual std::size_t **capacity** () const
- virtual std::size_t **position** () const
- virtual **Buffer** & **position** (std::size_t newPosition) throw (lang::exceptions::IllegalArgumentException)
Sets this buffer's position.
- virtual std::size_t **limit** () const
- virtual **Buffer** & **limit** (std::size_t newLimit) throw (lang::exceptions::IllegalArgumentException)
Sets this buffer's limit.
- virtual **Buffer** & **mark** ()
Sets this buffer's mark at its position.
- virtual **Buffer** & **reset** () throw (InvalidMarkException)
Resets this buffer's position to the previously-marked position.
- virtual **Buffer** & **clear** ()
Clears this buffer.
- virtual **Buffer** & **flip** ()
Flips this buffer.
- virtual **Buffer** & **rewind** ()
Rewinds this buffer.
- virtual std::size_t **remaining** () const
Returns the number of elements between the current position and the limit.
- virtual bool **hasRemaining** () const
Tells whether there are any elements between the current position and the limit.
- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.

Protected Attributes

- `std::size_t __position`
- `std::size_t __capacity`
- `std::size_t __limit`
- `std::size_t __mark`
- `bool __markSet`

6.138.1 Detailed Description

A container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: * Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p.837) and a relative put operation throws a **BufferOverflowException** (p.834); in either case, no data is transferred. * Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p.1813) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values: $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

clear() (p. 805) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

flip() (p. 806) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

rewind() (p. 808) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 2685) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

6.138.2 Constructor & Destructor Documentation

6.138.2.1 `decaf::nio::Buffer::Buffer (std::size_t capacity)`

6.138.2.2 `decaf::nio::Buffer::Buffer (const Buffer & other)`

6.138.2.3 `virtual decaf::nio::Buffer::~~Buffer ()` [inline, virtual]

6.138.3 Member Function Documentation

6.138.3.1 `virtual std::size_t decaf::nio::Buffer::capacity () const` [inline, virtual]

Returns:

this buffer's capacity.

6.138.3.2 `virtual Buffer& decaf::nio::Buffer::clear ()` [virtual]

Clears this buffer. The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

Returns:

a reference to this buffer.

6.138.3.3 virtual Buffer& decaf::nio::Buffer::flip () [virtual]

Flips this buffer. The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header
in.read(buf); // Read data into rest of buffer
buf.flip(); // Flip buffer
out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

Returns:

a reference to this buffer.

6.138.3.4 virtual bool decaf::nio::Buffer::hasRemaining () const [inline, virtual]

Tells whether there are any elements between the current position and the limit.

Returns:

true if, and only if, there is at least one element remaining in this buffer

6.138.3.5 virtual bool decaf::nio::Buffer::isReadOnly () const [pure virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 884), `decaf::internal::nio::CharArrayBuffer` (p. 995), `decaf::internal::nio::DoubleArrayBuffer` (p. 1553), `decaf::internal::nio::FloatArrayBuffer` (p. 1663), `decaf::internal::nio::IntArrayBuffer` (p. 1749), `decaf::internal::nio::LongArrayBuffer` (p. 2076), `decaf::internal::nio::ShortArrayBuffer` (p. 2920), and `decaf::nio::ByteBuffer` (p. 928).

6.138.3.6 virtual Buffer& decaf::nio::Buffer::limit (std::size_t newLimit) throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's limit. If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

Parameters:

newLimit - The new limit value; must be no larger than this buffer's capacity

Returns:

A reference to This buffer

Exceptions:

IllegalArgumentException if preconditions on the new pos don't hold.

6.138.3.7 virtual std::size_t decaf::nio::Buffer::limit () const [inline, virtual]

Returns:

this buffers Limit

6.138.3.8 virtual Buffer& decaf::nio::Buffer::mark () [virtual]

Sets this buffer's mark at its position.

Returns:

a reference to this buffer.

6.138.3.9 virtual Buffer& decaf::nio::Buffer::position (std::size_t *newPosition*)
throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's position. If the mark is defined and larger than the new position then it is discarded.

Parameters:

newPosition - the new postion in the buffer to set.

Returns:

A reference to This buffer

Exceptions:

IllegalArgumentException if preconditions on the new pos don't hold.

6.138.3.10 virtual std::size_t decaf::nio::Buffer::position () const [inline, virtual]

Returns:

the current position in the buffer

6.138.3.11 `virtual std::size_t decaf::nio::Buffer::remaining () const [inline, virtual]`

Returns the number of elements between the current position and the limit.

Returns:

The number of elements remaining in this buffer

6.138.3.12 `virtual Buffer& decaf::nio::Buffer::reset () throw (InvalidMarkException) [virtual]`

Resets this buffer's position to the previously-marked position.

Returns:

a reference to this buffer.

Exceptions:

InvalidMarkException (p. 1813) - If the mark has not been set

6.138.3.13 `virtual Buffer& decaf::nio::Buffer::rewind () [virtual]`

Rewinds this buffer. The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); // Copy data into array
```

Returns:

a reference to this buffer.

6.138.4 Field Documentation

6.138.4.1 `std::size_t decaf::nio::Buffer::_capacity [protected]`

6.138.4.2 `std::size_t decaf::nio::Buffer::_limit [protected]`

6.138.4.3 `std::size_t decaf::nio::Buffer::_mark [protected]`

6.138.4.4 `bool decaf::nio::Buffer::_markSet [protected]`

6.138.4.5 `std::size_t decaf::nio::Buffer::_position [mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/Buffer.h`

6.139 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

#include <src/main/decaf/io/BufferedInputStream.h> Inheritance diagram for decaf::io::BufferedInputStream:

Public Member Functions

- **BufferedInputStream** (**InputStream** *stream, bool **own**=false)
Constructor.
- **BufferedInputStream** (**InputStream** *stream, std::size_t bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException)
Constructor.
- virtual ~**BufferedInputStream** ()
- virtual std::size_t **available** () const throw (IOException)
Indicates the number of bytes available.
- virtual void **close** () throw (IOException)
*Close this **BufferedInputStream** (p. 809).*
- virtual int **read** () throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.

6.139.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

6.139.2 Constructor & Destructor Documentation

6.139.2.1 `decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, bool own = false)`

Constructor.

Parameters:

stream The target input stream.

own indicates if we own the stream object, defaults to false

6.139.2.2 `decaf::io::BufferedInputStream::BufferedInputStream (InputStream * stream, std::size_t bufferSize, bool own = false) throw (lang::exceptions::IllegalArgumentException)`

Constructor.

Parameters:

stream the target input stream

bufferSize the size for the internal buffer.

own indicates if we own the stream object, defaults to false.

Exceptions:

IllegalArgumentException is the size is zero.

6.139.2.3 `virtual decaf::io::BufferedInputStream::~~BufferedInputStream ()` [virtual]

6.139.3 Member Function Documentation

6.139.3.1 `virtual std::size_t decaf::io::BufferedInputStream::available () const` `throw (IOException)` [inline, virtual]

Indicates the number of bytes available.

Returns:

the sum of the amount of data available in the buffer and the data available on the target input stream.

Reimplemented from `decaf::io::FilterInputStream` (p.1633).

6.139.3.2 virtual void decaf::io::BufferedInputStream::close () throw (IOException) [virtual]

Close this **BufferedInputStream** (p. 809). This implementation closes the target stream and releases any resources associated with it.

Exceptions:

IOException (p. 1820) If an error occurs attempting to close this stream.

Reimplemented from **decaf::io::FilterInputStream** (p. 1633).

6.139.3.3 virtual void decaf::io::BufferedInputStream::mark (int readLimit) [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p. 1634).

6.139.3.4 virtual bool decaf::io::BufferedInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1634).

6.139.3.5 virtual int decaf::io::BufferedInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer. Blocks until the requested number of bytes are available.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException if buffer is NULL

Reimplemented from **decaf::io::FilterInputStream** (p.1635).

6.139.3.6 `virtual int decaf::io::BufferedInputStream::read () throw (IOException)`
[virtual]

Reads a single byte from the buffer. Blocks until the data is available.

Returns:

The next byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p.1636).

6.139.3.7 `virtual void decaf::io::BufferedInputStream::reset () throw (IOException)`
[inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p.1820) might be thrown. * If such an **IOException** (p.1820) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p.1820). * If an **IOException** (p.1820) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException (p. 1820)

Reimplemented from **decaf::io::FilterInputStream** (p.1636).

6.139.3.8 `virtual std::size_t decaf::io::BufferedInputStream::skip`
(std::size_t *num*) throw (io::IOException,
lang::exceptions::UnsupportedOperationException)
[virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from

any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If *n* is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1740) creates a byte array and then repeatedly reads into it until *n* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException (p. 1820) if an error occurs

Reimplemented from **decaf::io::FilterInputStream** (p. 1637).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedInputStream.h**

6.140 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

#include <src/main/decaf/io/BufferedOutputStream.h> Inheritance diagram for decaf::io::BufferedOutputStream:

Public Member Functions

- **BufferedOutputStream** (**OutputStream** *stream, bool **own**=false)
Constructor.
- **BufferedOutputStream** (**OutputStream** *stream, std::size_t bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException)
Constructor.
- virtual ~**BufferedOutputStream** ()
- virtual void **write** (unsigned char c) throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (IOException)
Invokes flush on the target output stream.
- void **close** () throw (io::IOException)
Invokes close on the target output stream.

6.140.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

6.140.2 Constructor & Destructor Documentation

6.140.2.1 decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, bool own = false)

Constructor.

Parameters:

- stream* The target output stream.
- own* Indicates if this class owns the stream pointer.

6.140.2.2 `decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, std::size_t bufferSize, bool own = false) throw (lang::exceptions::IllegalArgumentException)`

Constructor.

Parameters:

- stream* The target output stream.
- bufferSize* The size for the internal buffer.
- own* Indicates if this class owns the stream pointer.

6.140.2.3 `virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream () [virtual]`

6.140.3 Member Function Documentation

6.140.3.1 `void decaf::io::BufferedOutputStream::close () throw (io::IOException) [virtual]`

Invokes close on the target output stream.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1642).

6.140.3.2 `virtual void decaf::io::BufferedOutputStream::flush () throw (IOException) [virtual]`

Invokes flush on the target output stream.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1642).

6.140.3.3 `virtual void decaf::io::BufferedOutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException) [virtual]`

Writes an array of bytes to the output stream.

Parameters:

- buffer* The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1645).

6.140.3.4 `virtual void decaf::io::BufferedOutputStream::write (const std::vector< unsigned char > & buffer) throw (IOException) [virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1646).

6.140.3.5 `virtual void decaf::io::BufferedOutputStream::write (unsigned char c) throw (IOException) [virtual]`

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1646).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedOutputStream.h`

6.141 decaf::net::BufferedSocket Class Reference

Buffered **Socket** (p.2964) class that wraps a **Socket** (p.2964) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.

#include <src/main/decaf/net/BufferedSocket.h> Inheritance diagram for decaf::net::BufferedSocket:

Public Member Functions

- **BufferedSocket** (**Socket** *socket, int inputBufferSize=1000, int outputBufferSize=1000, bool own=true)
Constructs a new Buffered socket object.
- virtual ~**BufferedSocket** ()
- virtual void **connect** (const char *host, int port) throw (**SocketException**)
Connects to the specified destination.
- virtual void **close** () throw (**decaf::io::IOException**)
Closes this object and deallocates the appropriate resources.
- virtual bool **isConnected** () const
Indicates whether or not this socket is connected to a destination.
- virtual **io::InputStream** * **getInputStream** ()
Gets the InputStream for this socket.
- virtual **io::OutputStream** * **getOutputStream** ()
Gets the OutputStream for this socket.
- virtual int **getSoLinger** () const throw (**SocketException**)
Gets the linger time.
- virtual void **setSoLinger** (int linger) throw (**SocketException**)
Sets the linger time.
- virtual bool **getKeepAlive** () const throw (**SocketException**)
Gets the keep alive flag.
- virtual void **setKeepAlive** (bool keepAlive) throw (**SocketException**)
Enables/disables the keep alive flag.
- virtual int **getReceiveBufferSize** () const throw (**SocketException**)
Gets the receive buffer size.
- virtual void **setReceiveBufferSize** (int size) throw (**SocketException**)
Sets the receive buffer size.

- virtual bool **getReuseAddress** () const throw (SocketException)
Gets the reuse address flag.
- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
Sets the reuse address flag.
- virtual int **getSendBufferSize** () const throw (SocketException)
Gets the send buffer size.
- virtual void **setSendBufferSize** (int size) throw (SocketException)
Sets the send buffer size.
- virtual int **getSoTimeout** () const throw (SocketException)
Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout) throw (SocketException)
Sets the timeout for socket operations.

6.141.1 Detailed Description

Buffered **Socket** (p.2964) class that wraps a **Socket** (p.2964) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.

6.141.2 Constructor & Destructor Documentation

- 6.141.2.1** `decaf::net::BufferedSocket::BufferedSocket (Socket * socket, int inputBufferSize = 1000, int outputBufferSize = 1000, bool own = true)`

Constructs a new Buffered socket object.

Parameters:

socket the socket to buffer
inputBufferSize size of the input buffer
outputBufferSize size of the output buffer
own does this object own the passed socket

- 6.141.2.2** `virtual decaf::net::BufferedSocket::~~BufferedSocket ()` [virtual]

6.141.3 Member Function Documentation

- 6.141.3.1** `virtual void decaf::net::BufferedSocket::close () throw (decaf::io::IOException)` [virtual]

Closes this object and deallocates the appropriate resources.

Exceptions:

IOException

Implements **decaf::io::Closeable** (p.1019).

6.141.3.2 virtual void decaf::net::BufferedSocket::connect (const char * *host*, int *port*) throw (SocketException) [virtual]

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions:

IOException Thrown if a failure occurred in the connect.

Implements **decaf::net::Socket** (p. 2965).

6.141.3.3 virtual io::InputStream* decaf::net::BufferedSocket::getInputStream () [inline, virtual]

Gets the InputStream for this socket.

Returns:

The InputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2966).

6.141.3.4 virtual bool decaf::net::BufferedSocket::getKeepAlive () const throw (SocketException) [inline, virtual]

Gets the keep alive flag.

Returns:

True if keep alive is enabled.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2966).

References **decaf::net::Socket::getKeepAlive()**.

6.141.3.5 virtual io::OutputStream* decaf::net::BufferedSocket::getOutputStream () [inline, virtual]

Gets the OutputStream for this socket.

Returns:

the OutputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2966).

6.141.3.6 `virtual int decaf::net::BufferedSocket::getReceiveBufferSize () const throw (SocketException) [inline, virtual]`

Gets the receive buffer size.

Returns:

the receive buffer size in bytes.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2966).

References `decaf::net::Socket::getReceiveBufferSize()`.

6.141.3.7 `virtual bool decaf::net::BufferedSocket::getReuseAddress () const throw (SocketException) [inline, virtual]`

Gets the reuse address flag.

Returns:

True if the address can be reused.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2967).

References `decaf::net::Socket::getReuseAddress()`.

6.141.3.8 `virtual int decaf::net::BufferedSocket::getSendBufferSize () const throw (SocketException) [inline, virtual]`

Gets the send buffer size.

Returns:

the size in bytes of the send buffer.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2967).

References `decaf::net::Socket::getSendBufferSize()`.

6.141.3.9 `virtual int decaf::net::BufferedSocket::getSoLinger () const throw (SocketException) [inline, virtual]`

Gets the linger time.

Returns:

The linger time in seconds.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2967).

References **decaf::net::Socket::getSoLinger()**.

6.141.3.10 virtual int decaf::net::BufferedSocket::getSoTimeout () const throw (SocketException) [inline, virtual]

Gets the timeout for socket operations.

Returns:

The timeout in milliseconds for socket operations.

Exceptions:

SocketException (p. 2972) Thrown if unable to retrieve the information.

Implements **decaf::net::Socket** (p. 2967).

References **decaf::net::Socket::getSoTimeout()**.

6.141.3.11 virtual bool decaf::net::BufferedSocket::isConnected () const [inline, virtual]

Indicates whether or not this socket is connected to a destination.

Returns:

true if connected

Implements **decaf::net::Socket** (p. 2968).

References **decaf::net::Socket::isConnected()**.

6.141.3.12 virtual void decaf::net::BufferedSocket::setKeepAlive (bool *keepAlive*) throw (SocketException) [inline, virtual]

Enables/disables the keep alive flag.

Parameters:

keepAlive If true, enables the flag.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2968).

References **decaf::net::Socket::setKeepAlive()**.

6.141.3.13 `virtual void decaf::net::BufferedSocket::setReceiveBufferSize (int size)
throw (SocketException)` [inline, virtual]

Sets the receive buffer size.

Parameters:

size Number of bytes to set the receive buffer to.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2968).

References `decaf::net::Socket::setReceiveBufferSize()`.

6.141.3.14 `virtual void decaf::net::BufferedSocket::setReuseAddress (bool reuse)
throw (SocketException)` [inline, virtual]

Sets the reuse address flag.

Parameters:

reuse If true, sets the flag.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2969).

References `decaf::net::Socket::setReuseAddress()`.

6.141.3.15 `virtual void decaf::net::BufferedSocket::setSendBufferSize (int size)
throw (SocketException)` [inline, virtual]

Sets the send buffer size.

Parameters:

size The number of bytes to set the send buffer to.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2969).

References `decaf::net::Socket::setSendBufferSize()`.

6.141.3.16 `virtual void decaf::net::BufferedSocket::setSoLinger (int linger) throw (SocketException)` [inline, virtual]

Sets the linger time.

Parameters:

linger The linger time in seconds. If 0, linger is off.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2969).

References **decaf::net::Socket::setSoLinger()**.

**6.141.3.17 virtual void decaf::net::BufferedSocket::setSoTimeout (int *timeout*)
throw (SocketException) [inline, virtual]**

Sets the timeout for socket operations.

Parameters:

timeout The timeout in milliseconds for socket operations.

Exceptions:

SocketException (p. 2972) Thrown if unable to set the information.

Implements **decaf::net::Socket** (p. 2969).

References **decaf::net::Socket::setSoTimeout()**.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BufferedSocket.h`

6.142 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p.128) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

Public Member Functions

- virtual **~BufferFactory** ()

Static Public Member Functions

- static **decaf::nio::ByteBuffer * createByteBuffer** (std::size_t capacity)
Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ByteBuffer * createByteBuffer** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new ByteBuffer.
- static **decaf::nio::ByteBuffer * createByteBuffer** (std::vector< unsigned char > &buffer)
Wraps the passed STL Byte Vector in a ByteBuffer.
- static **decaf::nio::CharBuffer * createCharBuffer** (std::size_t capacity)
Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::CharBuffer * createCharBuffer** (char *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new CharBuffer.
- static **decaf::nio::CharBuffer * createCharBuffer** (std::vector< char > &buffer)
Wraps the passed STL Byte Vector in a CharBuffer.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (std::size_t capacity)
Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (double *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new DoubleBuffer.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (std::vector< double > &buffer)
Wraps the passed STL Double Vector in a DoubleBuffer.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (std::size_t capacity)
Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::FloatBuffer * createFloatBuffer** (float *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new FloatBuffer.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (std::vector< float > &buffer)
Wraps the passed STL Float Vector in a FloatBuffer.
- static **decaf::nio::LongBuffer * createLongBuffer** (std::size_t capacity)
Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::LongBuffer * createLongBuffer** (long long *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new LongBuffer.
- static **decaf::nio::LongBuffer * createLongBuffer** (std::vector< long long > &buffer)
Wraps the passed STL Long Vector in a LongBuffer.
- static **decaf::nio::IntBuffer * createIntBuffer** (std::size_t capacity)
Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::IntBuffer * createIntBuffer** (int *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new IntBuffer.
- static **decaf::nio::IntBuffer * createIntBuffer** (std::vector< int > &buffer)
Wraps the passed STL int Vector in a IntBuffer.
- static **decaf::nio::ShortBuffer * createShortBuffer** (std::size_t capacity)
Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ShortBuffer * createShortBuffer** (short *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new ShortBuffer.
- static **decaf::nio::ShortBuffer * createShortBuffer** (std::vector< short > &buffer)
Wraps the passed STL Short Vector in a ShortBuffer.

6.142.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p.128) package to create the various default version of the NIO interfaces.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 `virtual decaf::internal::nio::BufferFactory::~~BufferFactory () [inline, virtual]`

6.142.3 Member Function Documentation

6.142.3.1 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::vector< unsigned char > & buffer) [static]`

Wraps the passed STL Byte Vector in a ByteBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new ByteBuffer that is backed by *buffer*, caller owns.

6.142.3.2 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (unsigned char * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new ByteBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new ByteBuffer that is backed by *buffer*, caller owns.

6.142.3.3 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::size_t capacity) [static]`

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated ByteBuffer which the caller owns.

6.142.3.4 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::vector< char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a CharBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new CharBuffer that is backed by *buffer*, caller owns.

6.142.3.5 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new CharBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new CharBuffer that is backed by *buffer*, caller owns.

6.142.3.6 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::size_t *capacity*) [static]

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated CharBuffer which the caller owns.

6.142.3.7 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::vector< double > & buffer) [static]`

Wraps the passed STL Double Vector in a DoubleBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.8 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (double * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new DoubleBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.9 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::size_t capacity) [static]`

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.142.3.10 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::vector< float > & buffer) [static]`

Wraps the passed STL Float Vector in a FloatBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.11 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (float * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new FloatBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.12 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::size_t capacity) [static]`

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.142.3.13 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::vector< int > & buffer) [static]`

Wraps the passed STL int Vector in a IntBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.14 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new IntBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.15 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::size_t capacity) [static]`

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.142.3.16 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::vector< long long > & buffer) [static]`

Wraps the passed STL Long Vector in a LongBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.17 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (long long * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new LongBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.18 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::size_t capacity) [static]`

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.142.3.19 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::vector< short > & buffer) [static]`

Wraps the passed STL Short Vector in a ShortBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.20 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (short * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new ShortBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.21 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::size_t capacity) [static]`

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/BufferFactory.h`

6.143 decaf::nio::BufferOverflowException Class Reference

#include <src/main/decaf/nio/BufferOverflowException.h> Inheritance diagram for decaf::nio::BufferOverflowException:

Public Member Functions

- **BufferOverflowException** () throw ()
Default Constructor.
- **BufferOverflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const BufferOverflowException &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BufferOverflowException** (const std::exception *cause) throw ()
Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **BufferOverflowException** * clone () const
Clones this exception.
- virtual ~**BufferOverflowException** () throw ()

6.143.1 Constructor & Destructor Documentation

6.143.1.1 decaf::nio::BufferOverflowException::BufferOverflowException () throw () [inline]

Default Constructor.

6.143.1.2 decaf::nio::BufferOverflowException::BufferOverflowException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.143.1.3 decaf::nio::BufferOverflowException::BufferOverflowException (const BufferOverflowException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.143.1.4 decaf::nio::BufferOverflowException::BufferOverflowException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.143.1.5 decaf::nio::BufferOverflowException::BufferOverflowException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.143.1.6 decaf::nio::BufferOverflowException::BufferOverflowException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.143.1.7 `virtual decaf::nio::BufferOverflowException::~~BufferOverflowException
() throw () [inline, virtual]`

6.143.2 Member Function Documentation

6.143.2.1 `virtual BufferOverflowException* de-
caf::nio::BufferOverflowException::clone () const [inline,
virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferOverflowException.h`

6.144 decaf::nio::BufferUnderflowException Class Reference

#include <src/main/decaf/nio/BufferUnderflowException.h> Inheritance diagram for decaf::nio::BufferUnderflowException:

Public Member Functions

- **BufferUnderflowException** () throw ()
Default Constructor.
- **BufferUnderflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferUnderflowException** (const BufferUnderflowException &ex) throw ()
Copy Constructor.
- **BufferUnderflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BufferUnderflowException** (const std::exception *cause) throw ()
Constructor.
- **BufferUnderflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **BufferUnderflowException** * clone () const
Clones this exception.
- virtual ~**BufferUnderflowException** () throw ()

6.144.1 Constructor & Destructor Documentation

6.144.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException () throw () [inline]

Default Constructor.

6.144.1.2 decaf::nio::BufferUnderflowException::BufferUnderflowException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.144.1.3 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const BufferUnderflowException & ex) throw ()` [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.144.1.4 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.144.1.5 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const std::exception * cause) throw ()` [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.144.1.6 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.144.1.7 **virtual**
decaf::nio::BufferUnderflowException::~~BufferUnderflowException ()
throw () [inline, virtual]

6.144.2 Member Function Documentation

6.144.2.1 **virtual BufferUnderflowException* de-**
caf::nio::BufferUnderflowException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferUnderflowException.h`

6.145 decaf::lang::Byte Class Reference

#include <src/main/decaf/lang/Byte.h> Inheritance diagram for decaf::lang::Byte:

Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Byte** ()
- virtual int **compareTo** (const **Byte** &c) const
*Compares this **Byte** (p. 840) instance with another.*
- virtual bool **operator==** (const **Byte** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Byte** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const unsigned char &c) const
*Compares this **Byte** (p. 840) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const unsigned char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Byte** (p. 840).*
- static unsigned char **parseByte** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed unsigned char in the radix specified by the second argument.
- static unsigned char **parseByte** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal unsigned char.
- static **Byte valueOf** (unsigned char value)
*Returns a **Character** (p. 982) instance representing the specified char value.*
- static **Byte valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Byte** (p. 840) object holding the value given by the specified std::string.*
- static **Byte valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Byte** (p. 840) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const unsigned char **MIN_VALUE** = 0x7F
The minimum value that a unsigned char can take on.
- static const unsigned char **MAX_VALUE** = 0x80
The maximum value that a unsigned char can take on.
- static const int **SIZE** = 8
The size of the primitive charactor in bits.

6.145.1 Constructor & Destructor Documentation

6.145.1.1 decaf::lang::Byte::Byte (unsigned char *value*)

Parameters:

value - the primitive value to wrap

6.145.1.2 `decaf::lang::Byte::Byte (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters:

value - the string to convert to an unsigned char

Exceptions:

NumberFormatException

6.145.1.3 `virtual decaf::lang::Byte::~~Byte () [inline, virtual]`

6.145.2 Member Function Documentation

6.145.2.1 `virtual unsigned char decaf::lang::Byte::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2432).

6.145.2.2 `virtual int decaf::lang::Byte::compareTo (const unsigned char & c) const [inline, virtual]`

Compares this **Byte** (p. 840) instance with a char type.

Parameters:

c - the char instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< unsigned char >` (p. 1083).

6.145.2.3 `virtual int decaf::lang::Byte::compareTo (const Byte & c) const [inline, virtual]`

Compares this **Byte** (p. 840) instance with another.

Parameters:

c - the **Byte** (p. 840) instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Byte** > (p.1083).

6.145.2.4 static Byte decaf::lang::Byte::decode (const std::string & value) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Byte** (p.840). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p.846) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Byte** (p.840) object containing the decoded value

Exceptions:

NumberFormatException if the string is not formatted correctly.

6.145.2.5 virtual double decaf::lang::Byte::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p.2433).

6.145.2.6 bool decaf::lang::Byte::equals (const unsigned char & c) const [inline, virtual]**Returns:**

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable**< **unsigned char** > (p.1084).

6.145.2.7 `bool decaf::lang::Byte::equals (const Byte & c) const [inline, virtual]`

Returns:

true if the two **Byte** (p. 840) Objects have the same value.

Implements **decaf::lang::Comparable**< **Byte** > (p.1084).

6.145.2.8 `virtual float decaf::lang::Byte::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.145.2.9 `virtual int decaf::lang::Byte::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.145.2.10 `virtual long long decaf::lang::Byte::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.145.2.11 `virtual bool decaf::lang::Byte::operator< (const unsigned char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 1084).

6.145.2.12 virtual bool decaf::lang::Byte::operator< (const Byte & c) const
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Byte >** (p.1084).

6.145.2.13 virtual bool decaf::lang::Byte::operator== (const unsigned char & c)
const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p.1085).

6.145.2.14 virtual bool decaf::lang::Byte::operator== (const Byte & c) const
[inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Byte >** (p.1085).

6.145.2.15 static unsigned char decaf::lang::Byte::parseByte (const std::string & s)
throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal unsigned char. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseByte(const std::string, int) method.

Parameters:

s - String to convert to a unsigned char

Returns:

the converted unsigned char value

Exceptions:

NumberFormatException if the string is not a unsigned char.

6.145.2.16 `static unsigned char decaf::lang::Byte::parseByte (const std::string & s, int radix) throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed unsigned char in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 985) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type *NumberFormatException* is thrown if any of the following situations occurs:
 * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 989) or larger than **Character::MAX_RADIX** (p. 989). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type unsigned char.

Parameters:

s - the String containing the unsigned char to be parsed

radix - the radix to be used while parsing s

Returns:

the unsigned char represented by the string argument in the specified radix.

Exceptions:

NumberFormatException - If String does not contain a parsable unsigned char.

6.145.2.17 `virtual short decaf::lang::Byte::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2434).

6.145.2.18 `static std::string decaf::lang::Byte::toString (unsigned char value) [static]`

Returns:

a string representing the primitive value as Base 10

6.145.2.19 `std::string decaf::lang::Byte::toString () const`**Returns:**

this **Byte** (p. 840) Object as a String Representation

6.145.2.20 `static Byte decaf::lang::Byte::valueOf (const std::string & value, int radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Byte** (p. 840) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parseByte(std::string, int)` method. The result is a **Byte** (p. 840) object that represents the unsigned char value specified by the string.

Parameters:

value - `std::string` to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Byte** (p. 840) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid unsigned char.

6.145.2.21 `static Byte decaf::lang::Byte::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]`

Returns a **Byte** (p. 840) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte(std::string)` method. The result is a **Byte** (p. 840) object that represents the unsigned char value specified by the string.

Parameters:

value - `std::string` to parse as base 10

Returns:

new **Byte** (p. 840) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal unsigned char.

6.145.2.22 `static Byte decaf::lang::Byte::valueOf (unsigned char value) [inline, static]`

Returns a **Character** (p. 982) instance representing the specified char value.

Parameters:

value - the primitive char to wrap.

Returns:

a new **Character** (p. 982) instance that wraps this value.

6.145.3 Field Documentation

6.145.3.1 `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80` [static]

The maximum value that a unsigned char can take on.

6.145.3.2 `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F` [static]

The minimum value that a unsigned char can take on.

6.145.3.3 `const int decaf::lang::Byte::SIZE = 8` [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

6.146 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

#include <src/main/decaf/internal/util/ByteArrayAdapter.h> Inheritance diagram for decaf::internal::util::ByteArrayAdapter:

Data Structures

- union **Array**

Public Member Functions

- **ByteArrayAdapter** (std::size_t capacity)
Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayAdapter** (unsigned char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (double *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (float *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (long long *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (int *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (short *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- virtual ~**ByteArrayAdapter** ()

- virtual std::size_t **getCapacity** () const
Gets the capacity of the underlying array.
- virtual std::size_t **getCharCapacity** () const
Gets the capacity of the underlying array as if it contains chars.
- virtual std::size_t **getDoubleCapacity** () const
Gets the capacity of the underlying array as if it contains doubles.
- virtual std::size_t **getFloatCapacity** () const
Gets the capacity of the underlying array as if it contains doubles.
- virtual std::size_t **getLongCapacity** () const
Gets the capacity of the underlying array as if it contains doubles.
- virtual std::size_t **getIntCapacity** () const
Gets the capacity of the underlying array as if it contains ints.
- virtual std::size_t **getShortCapacity** () const
Gets the capacity of the underlying array as if it contains shorts.
- virtual unsigned char * **getByteArray** ()
Gets the pointer to the array we are wrapping.
- virtual char * **getCharArray** ()
Gets the pointer to the array we are wrapping.
- virtual short * **getShortArray** ()
Gets the pointer to the array we are wrapping.
- virtual int * **getIntArray** ()
Gets the pointer to the array we are wrapping.
- virtual long long * **getLongArray** ()
Gets the pointer to the array we are wrapping.
- virtual double * **getDoubleArray** ()
Gets the pointer to the array we are wrapping.
- virtual float * **getFloatArray** ()
Gets the pointer to the array we are wrapping.
- virtual void **read** (unsigned char *buffer, std::size_t offset, std::size_t length) const throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException)
Reads from the Byte array starting at the specified offset and reading the specified length.
- virtual void **write** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException)
Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

- virtual void **resize** (std::size_t capacity) throw (lang::exceptions::InvalidStateException)
Resizes the underlying array to the new given capacity, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.
- virtual void **clear** ()
Clear all data from that Array, setting the underlying bytes to zero.
- unsigned char & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Allows the **ByteArrayAdapter** (p. 849) to be indexed as a standard array.*
- const unsigned char & **operator[]** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual unsigned char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual char **getChar** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads one byte at the given index and returns it.
- virtual double **getDouble** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual double **getDoubleAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given byte index and returns it.
- virtual float **getFloat** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual float **getFloatAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given byte index and returns it.
- virtual long long **getLong** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual long long **getLongAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given byte index and returns it.
- virtual int **getInt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.

- virtual int **getIntAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given byte index and returns it.
- virtual short **getShort** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given index and returns it.
- virtual short **getShortAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given byte index and returns it.
- virtual **ByteArrayAdapter** & **put** (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes the given byte into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putChar** (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putDouble** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putDoubleAt** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putFloat** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putFloatAt** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putLong** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putLongAt** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putInt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putIntAt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putShort** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)

Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putShortAt** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)

Writes two bytes containing the given value, into this buffer at the given byte index.

6.146.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data. All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

6.146.2 Constructor & Destructor Documentation

6.146.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (std::size_t capacity)

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.146.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.3 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.4 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (double * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.5 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (float * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.6 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (long long * *array*, std::size_t *capacity*, bool *own* = false) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.7 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int * *array*, std::size_t *capacity*, bool *own* = false) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.8 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (short * *array*, std::size_t *capacity*, bool *own* = false) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.146.2.9 virtual decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter ()
[virtual]

6.146.3 Member Function Documentation

6.146.3.1 virtual void decaf::internal::util::ByteArrayAdapter::clear () [virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.146.3.2 virtual unsigned char decaf::internal::util::ByteArrayAdapter::get
(std::size_t *index*) const throw (
lang::exceptions::IndexOutOfBoundsException)
[virtual]

Absolute get method. Reads the byte at the given index.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the byte that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

6.146.3.3 virtual unsigned char* de-
caaf::internal::util::ByteArrayAdapter::getByteArray ()
[inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 849) objects that point to this array.

Returns:

an unsigned char* pointer to the array this object wraps.

6.146.3.4 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getCapacity ()
const [inline, virtual]

Gets the capacity of the underlying array.

Returns:

the size the array.

6.146.3.5 virtual char decaf::internal::util::ByteArrayAdapter::getChar (std::size_t
index) const throw (lang::exceptions::IndexOutOfBoundsException)
[inline, virtual]

Reads one byte at the given index and returns it.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the char at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

6.146.3.6 virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray ()
[inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.849) objects that point to this array.

Returns:

an char* pointer to the array this object wraps.

6.146.3.7 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getCharCapacity ()
const [inline, virtual]

Gets the capacity of the underlying array as if it contains chars.

Returns:

the size the array.

6.146.3.8 virtual double decaf::internal::util::ByteArrayAdapter::getDouble
(std::size_t *index*) const throw (
lang::exceptions::IndexOutOfBoundsException)
[virtual]

Reads eight bytes at the given index and returns it. The index is a relative to the size of the type to be read, in otherwords when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.9 `virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray
() [inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.849) objects that point to this array.

Returns:

an double* pointer to the array this object wraps.

6.146.3.10 `virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt
(std::size_t index) const throw (
lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Reads eight bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.11 `virtual std::size_t de-
cafe::internal::util::ByteArrayAdapter::getDoubleCapacity
() const [inline, virtual]`

Gets the capacity of the underlying array as if it contains doubles.

Returns:

the size the array.

6.146.3.12 `virtual float decaf::internal::util::ByteArrayAdapter::getFloat
(std::size_t index) const throw (
lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Reads four bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.13 virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray ()
[inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.849) objects that point to this array.

Returns:

an float* pointer to the array this object wraps.

6.146.3.14 virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt
(std::size_t *index*) const throw (
lang::exceptions::IndexOutOfBoundsException)
[virtual]

Reads four bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.15 virtual std::size_t de-
caf::internal::util::ByteArrayAdapter::getFloatCapacity ()
const [inline, virtual]

Gets the capacity of the underlying array as if it contains doubles.

Returns:

the size the array.

6.146.3.16 `virtual int decaf::internal::util::ByteArrayAdapter::getInt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads four bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.17 `virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.849) objects that point to this array.

Returns:

an int* pointer to the array this object wraps.

6.146.3.18 `virtual int decaf::internal::util::ByteArrayAdapter::getIntAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads four bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.19 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getIntCapacity () const [inline, virtual]`

Gets the capacity of the underlying array as if it contains ints.

Returns:

the size the array.

6.146.3.20 `virtual long long decaf::internal::util::ByteArrayAdapter::getLong (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.21 `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray () [inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 849) objects that point to this array.

Returns:

an long long* pointer to the array this object wraps.

6.146.3.22 `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.23 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getLongCapacity () const [inline, virtual]`

Gets the capacity of the underlying array as if it contains doubles.

Returns:

the size the array.

6.146.3.24 `virtual short decaf::internal::util::ByteArrayAdapter::getShort (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads two bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.25 `virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray () [inline, virtual]`

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 849) objects that point to this array.

Returns:

an short* pointer to the array this object wraps.

6.146.3.26 virtual short decaf::internal::util::ByteArrayAdapter::getShortAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]

Reads two bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.27 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getShortCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains shorts.

Returns:

the size the array.

6.146.3.28 const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

6.146.3.29 unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (std::size_t *index*) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Allows the **ByteArrayAdapter** (p. 849) to be indexed as a standard array. calling the non const version allows the user to change the value at index

Parameters:

index - the position in the array to access

Exceptions:

IndexOutOfBoundsException

6.146.3.30 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Writes the given byte into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the byte to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.31 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putChar (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Writes one byte containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.32 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Writes eight bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.33 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.34 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]

Writes four bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.35 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putFloatAt (std::size_t index,
float value) throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.36 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putInt (std::size_t
index, int value) throw (lang::exceptions::IndexOutOfBoundsException
) [virtual]`

Writes four bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.37 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putIntAt (std::size_t index,
int value) throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.38 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.39 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.40 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Writes two bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.41 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.42 `virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char * buffer, std::size_t offset, std::size_t length) const throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException)` [virtual]

Reads from the Byte array starting at the specified offset and reading the specified length. If the length is greater than the capacity of this underlying byte array then an `BufferUnderflowException` is thrown.

Parameters:

buffer - the buffer to read data from this array into.

offset - position in this array to start reading from.

length - the amount of data to read from this array.

Exceptions:

NullPointerException if buffer is null

BufferUnderflowException if there is not enough data to read because the offset or the length is greater than the size of this array.

6.146.3.43 virtual void decaf::internal::util::ByteArrayAdapter::resize (std::size_t capacity) throw (lang::exceptions::InvalidStateException) [virtual]

Resizes the underlying array to the new given capacity, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved. A **ByteArrayAdapter** (p. 849) can only be resized when it owns the underlying array, if it does not then it will throw an *InvalidStateException*.

Parameters:

capacity - the new capacity of the array.

Exceptions:

InvalidStateException if this object does not own the buffer.

6.146.3.44 virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char * buffer, std::size_t offset, std::size_t length) throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException) [virtual]

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array. . If the length is greater than the capacity of this underlying byte array then an *BufferOverflowException* is thrown.

Parameters:

buffer - the buffer to write get data written into this array.

offset - position in this array to start writing from.

length - the amount of data to write to this array.

Exceptions:

NullPointerException if buffer is null

BufferOverflowException if the amount of data to be written to this array or the offset given are larger than this array's capacity.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/ByteArrayAdapter.h

6.147 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:.

#include <src/main/decaf/internal/nio/ByteBuffer.h> Inheritance diagram for decaf::internal::nio::ByteBuffer:

Public Member Functions

- **ByteBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **ByteBuffer** (p. 870) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ByteBuffer** (unsigned char *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **ByteBuffer** (p. 870) object that wraps the given array.*
- **ByteBuffer** (ByteBufferPerspective &array, std::size_t offset, std::size_t length, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteBufferPerspective** (p. 908) and start at the given offset.*
- **ByteBuffer** (const ByteBuffer &other)
*Create a **ByteBuffer** (p. 870) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~ByteBuffer ()
- virtual bool isReadOnly () const
Tells whether or not this buffer is read-only.
- virtual unsigned char * array () throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)
Returns the byte array that backs this buffer.
- virtual std::size_t arrayOffset () const throw (decaf::nio::ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)
Returns the offset within this buffer's backing array of the first element of the buffer.
- virtual bool hasArray () const
Tells whether or not this buffer is backed by an accessible byte array.
- virtual decaf::nio::CharBuffer * asCharBuffer () const
Creates a view of this byte buffer as a char buffer.
- virtual decaf::nio::DoubleBuffer * asDoubleBuffer () const
Creates a view of this byte buffer as a double buffer.
- virtual decaf::nio::FloatBuffer * asFloatBuffer () const

Creates a view of this byte buffer as a float buffer.

- virtual **decaf::nio::IntBuffer** * **asIntBuffer** () const
Creates a view of this byte buffer as a int buffer.
- virtual **decaf::nio::LongBuffer** * **asLongBuffer** () const
Creates a view of this byte buffer as a long buffer.
- virtual **decaf::nio::ShortBuffer** * **asShortBuffer** () const
Creates a view of this byte buffer as a short buffer.
- virtual **ByteBuffer** * **asReadOnlyBuffer** () const
Creates a new, read-only byte buffer that shares this buffer's content.
- virtual **ByteBuffer** & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual **ByteBuffer** * **duplicate** ()
Creates a new byte buffer that shares this buffer's content.
- virtual unsigned char **get** () const throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual unsigned char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual char **getChar** () throw (decaf::nio::BufferUnderflowException)
Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual char **getChar** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads one byte at the given index and returns it.
- virtual double **getDouble** () throw (decaf::nio::BufferUnderflowException)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual float **getFloat** () throw (decaf::nio::BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual float **getFloat** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual long long **getLong** () throw (decaf::nio::BufferUnderflowException)

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual long long **getLong** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads eight bytes at the given index and returns it.

- virtual int **getInt** () throw (decaf::nio::BufferUnderflowException)

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual int **getInt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads four bytes at the given index and returns it.

- virtual short **getShort** () throw (decaf::nio::BufferUnderflowException)

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

- virtual short **getShort** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads two bytes at the given index and returns it.

- virtual **ByteBuffer** & **put** (unsigned char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given byte into this buffer at the current position, and then increments the position.

- virtual **ByteBuffer** & **put** (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given byte into this buffer at the given index.

- virtual **ByteBuffer** & **putChar** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

- virtual **ByteBuffer** & **putChar** (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes one byte containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putDouble** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putDouble** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putFloat** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putFloat** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putLong** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putLong** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putInt** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putInt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putShort** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putShort** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** * **slice** () const

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 870) as Read-Only.*

6.147.1 Detailed Description

This class defines six categories of operations upon byte buffers:. 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer;

4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 882) float getFloat(int index) void **putFloat(float f)** (p. 888) void putFloat(int index, float f)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the FloatBuffer class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.147.2 Constructor & Destructor Documentation

6.147.2.1 `decaf::internal::nio::ByteBuffer::ByteBuffer (std::size_t capacity, bool readOnly = false)`

Creates a **ByteBuffer** (p. 870) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.147.2.2 decaf::internal::nio::ByteBuffer::ByteBuffer (unsigned char * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **ByteBuffer** (p. 870) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.147.2.3 decaf::internal::nio::ByteBuffer::ByteBuffer (ByteBufferPerspective & *array*, std::size_t *offset*, std::size_t *length*, bool *readOnly* = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteBufferPerspective** (p. 908) and start at the given offset. The capacity and limit of the new **ByteBuffer** (p. 870) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteBufferPerspective** (p. 908) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.147.2.4 decaf::internal::nio::ByteBuffer::ByteBuffer (const ByteBuffer & *other*)

Create a **ByteBuffer** (p. 870) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferPerspective** (p. 908) and when changes are made to that data it is reflected in both.

Parameters:

other - the **ByteBuffer** (p. 870) this one is to mirror.

6.147.2.5 `virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer ()`
[virtual]

6.147.3 Member Function Documentation

6.147.3.1 `virtual unsigned char* decaf::internal::nio::ByteBuffer::array`
`() throw (decaf::nio::ReadOnlyBufferException,`
`decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Returns the byte array that backs this buffer. Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The array that backs this buffer

Exceptions:

ReadOnlyBufferException - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implements `decaf::nio::ByteBuffer` (p. 918).

6.147.3.2 `virtual std::size_t decaf::internal::nio::ByteBuffer::arrayOffset`
`() const throw (decaf::nio::ReadOnlyBufferException,`
`lang::exceptions::UnsupportedOperationException)` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer. If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 876).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset within this buffer's array of the first element of the buffer

Exceptions:

ReadOnlyBufferException - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implements `decaf::nio::ByteBuffer` (p. 919).

6.147.3.3 `virtual decaf::nio::CharBuffer* de-`
`caf::internal::nio::ByteBuffer::asCharBuffer () const`
[inline, virtual]

Creates a view of this byte buffer as a char buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new Char Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p.919).

6.147.3.4 virtual decaf::nio::DoubleBuffer* decaf::internal::nio::ByteBuffer::asDoubleBuffer () const [inline, virtual]

Creates a view of this byte buffer as a double buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new double Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p.919).

6.147.3.5 virtual decaf::nio::FloatBuffer* decaf::internal::nio::ByteBuffer::asFloatBuffer () const [inline, virtual]

Creates a view of this byte buffer as a float buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new float Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p.920).

6.147.3.6 virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteBuffer::asIntBuffer () const [inline, virtual]

Creates a view of this byte buffer as a int buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new `int Buffer`, which the caller then owns.

Implements `decaf::nio::ByteBuffer` (p. 920).

6.147.3.7 `virtual decaf::nio::LongBuffer* decaf::internal::nio::ByteBuffer::asLongBuffer () const`
[inline, virtual]

Creates a view of this byte buffer as a long buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new long `Buffer`, which the caller then owns.

Implements `decaf::nio::ByteBuffer` (p. 920).

6.147.3.8 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asReadOnlyBuffer () const`
[virtual]

Creates a new, read-only byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only byte buffer which the caller then owns.

Implements `decaf::nio::ByteBuffer` (p. 921).

6.147.3.9 `virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteBuffer::asShortBuffer () const`
[inline, virtual]

Creates a view of this byte buffer as a short buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new short Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 921).

6.147.3.10 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **ByteBuffer** (p. 870)

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 921).

6.147.3.11 virtual ByteBuffer* decaf::internal::nio::ByteBuffer::duplicate () [virtual]

Creates a new byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new Byte Buffer which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 922).

6.147.3.12 virtual unsigned char decaf::internal::nio::ByteBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the byte at the given index.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the byte that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ByteBuffer** (p. 922).

6.147.3.13 `virtual unsigned char decaf::internal::nio::ByteBuffer::get () const throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method. Reads the byte at this buffer's current position, and then increments the position.

Returns:

The byte at the buffer's current position

Exceptions:

BufferUnderflowException - If the buffer's current position is not smaller than its limit

Implements **decaf::nio::ByteBuffer** (p. 923).

6.147.3.14 `virtual char decaf::internal::nio::ByteBuffer::getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Reads one byte at the given index and returns it.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the char at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ByteBuffer** (p. 924).

6.147.3.15 `virtual char decaf::internal::nio::ByteBuffer::getChar () throw (decaf::nio::BufferUnderflowException) [inline, virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns:

the next char in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 924).

6.147.3.16 virtual double decaf::internal::nio::ByteBuffer::getDouble (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 925).

6.147.3.17 virtual double decaf::internal::nio::ByteBuffer::getDouble () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next double in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 925).

6.147.3.18 virtual float decaf::internal::nio::ByteBuffer::getFloat (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 925).

6.147.3.19 **virtual float decaf::internal::nio::ByteArrayBuffer::getFloat () throw (decaf::nio::BufferUnderflowException) [virtual]**

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next float in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 926).

6.147.3.20 **virtual int decaf::internal::nio::ByteArrayBuffer::getInt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]**

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 926).

6.147.3.21 `virtual int decaf::internal::nio::ByteBuffer::getInt () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next int in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements `decaf::nio::ByteBuffer` (p. 926).

6.147.3.22 `virtual long long decaf::internal::nio::ByteBuffer::getLong (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements `decaf::nio::ByteBuffer` (p. 927).

6.147.3.23 `virtual long long decaf::internal::nio::ByteBuffer::getLong () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next long long in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements `decaf::nio::ByteBuffer` (p. 927).

6.147.3.24 `virtual short decaf::internal::nio::ByteBuffer::getShort (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads two bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements `decaf::nio::ByteBuffer` (p. 927).

6.147.3.25 `virtual short decaf::internal::nio::ByteBuffer::getShort () throw (decaf::nio::BufferUnderflowException)` [virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next short in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements `decaf::nio::ByteBuffer` (p. 928).

6.147.3.26 `virtual bool decaf::internal::nio::ByteBuffer::hasArray () const`
[inline, virtual]

Tells whether or not this buffer is backed by an accessible byte array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::ByteBuffer` (p. 928).

6.147.3.27 `virtual bool decaf::internal::nio::ByteBuffer::isReadOnly () const`
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 928).

6.147.3.28 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put
(std::size_t index, unsigned char value) throw
(lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given byte into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the byte to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 929).

6.147.3.29 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put
(unsigned char value) throw (decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters:

value - the byte value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 929).

6.147.3.30 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 931).

6.147.3.31 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 931).

6.147.3.32 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 932).

6.147.3.33 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (double *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 932).

6.147.3.34 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 933).

6.147.3.35 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (float *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]**

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 933).

6.147.3.36 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]**

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 933).

6.147.3.37 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements decaf::nio::ByteBuffer (p. 934).

6.147.3.38 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements decaf::nio::ByteBuffer (p. 934).

6.147.3.39 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (long long *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 935).

6.147.3.40 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes two bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 935).

6.147.3.41 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 935).

6.147.3.42 `virtual void decaf::internal::nio::ByteBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.147.3.43 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice () const [virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ByteBuffer** (p. 870) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 936).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

6.148 decaf::io::ByteArrayInputStream Class Reference

Simple implementation of **InputStream** (p.1740) that wraps around an STL Vector `std::vector<unsigned char>`.

`#include <src/main/decaf/io/ByteArrayInputStream.h>` Inheritance diagram for `decaf::io::ByteArrayInputStream`:

Public Member Functions

- **ByteArrayInputStream** ()
Default Constructor.
- **ByteArrayInputStream** (const std::vector< unsigned char > &buffer)
Creates the input stream and calls setBuffer with the specified buffer object.
- **ByteArrayInputStream** (const unsigned char *buffer, std::size_t bufferSize)
Constructor.
- virtual ~**ByteArrayInputStream** ()
- virtual void **setBuffer** (const std::vector< unsigned char > &buffer)
Sets the internal buffer.
- virtual void **setByteArray** (const unsigned char *buffer, std::size_t bufferSize)
Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual std::size_t **available** () const throw (IOException)
Indicates the number of bytes available.
- virtual int **read** () throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (io::IOException)
Closes the target input stream.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (IOException)

Resets the read index to the beginning of the byte array, unless mark has been called and the markLimit has not been exceeded, in which case the stream is reset to the marked position.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Protected Member Functions

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.148.1 Detailed Description

Simple implementation of **InputStream** (p.1740) that wraps around an STL Vector `std::vector<unsigned char>`. Closing a **ByteArrayInputStream** (p.892) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p.1820).

6.148.2 Constructor & Destructor Documentation

6.148.2.1 `decaf::io::ByteArrayInputStream::ByteArrayInputStream ()`

Default Constructor.

6.148.2.2 `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const std::vector< unsigned char > & buffer)`

Creates the input stream and calls `setBuffer` with the specified buffer object.

Parameters:

buffer The buffer to be used.

6.148.2.3 `decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * buffer, std::size_t bufferSize)`

Constructor.

Parameters:

buffer initial byte array to use to read from

bufferSize the size of the buffer

6.148.2.4 `virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream ()` [virtual]

6.148.3 Member Function Documentation

6.148.3.1 `virtual std::size_t decaf::io::ByteArrayInputStream::available () const throw (IOException)` [inline, virtual]

Indicates the number of bytes available.

Returns:

The number of bytes until the end of the internal buffer.

Implements `decaf::io::InputStream` (p. 1741).

6.148.3.2 `virtual void decaf::io::ByteArrayInputStream::close () throw (io::IOException)` [inline, virtual]

Closes the target input stream.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements `decaf::io::Closeable` (p. 1019).

6.148.3.3 `virtual void decaf::io::ByteArrayInputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.148.3.4 `virtual void decaf::io::ByteArrayInputStream::mark (int readLimit DECAF_UNUSED) [inline, virtual]`

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1741).

6.148.3.5 `virtual bool decaf::io::ByteArrayInputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1741).

6.148.3.6 `virtual void decaf::io::ByteArrayInputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, protected, virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

6.148.3.7 `virtual void decaf::io::ByteArrayInputStream::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
protected, virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3125).

6.148.3.8 `virtual int decaf::io::ByteArrayInputStream::read (unsigned char *
buffer, std::size_t offset, std::size_t bufferSize) throw (IOException,
lang::exceptions::NullPointerException) [virtual]`

Reads an array of bytes from the buffer.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements `decaf::io::InputStream` (p. 1742).

6.148.3.9 `virtual int decaf::io::ByteArrayInputStream::read () throw (IOException
) [virtual]`

Reads a single byte from the buffer.

Returns:

The next byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements `decaf::io::InputStream` (p. 1742).

6.148.3.10 virtual void decaf::io::ByteArrayInputStream::reset () throw (IOException) [virtual]

Resets the read index to the beginning of the byte array, unless mark has been called and the markLimit has not been exceeded, in which case the stream is reset to the marked position.

Implements **decaf::io::InputStream** (p. 1742).

6.148.3.11 virtual void decaf::io::ByteArrayInputStream::setBuffer (const std::vector< unsigned char > & buffer) [virtual]

Sets the internal buffer. The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

Parameters:

buffer The buffer to be used.

6.148.3.12 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * buffer, std::size_t bufferSize) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters:

buffer initial byte array to use to read from

bufferSize the size of the buffer

6.148.3.13 virtual std::size_t decaf::io::ByteArrayInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1740) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException (p. 1820) if an error occurs

Implements **decaf::io::InputStream** (p. 1743).

6.148.3.14 `virtual bool decaf::io::ByteArrayInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3126).

6.148.3.15 `virtual void decaf::io::ByteArrayInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3127).

6.148.3.16 `virtual void decaf::io::ByteArrayInputStream::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.148.3.17 `virtual void decaf::io::ByteArrayInputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.148.3.18 `virtual void decaf::io::ByteArrayInputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayInputStream.h`

6.149 decaf::io::ByteArrayOutputStream Class Reference

#include <src/main/decaf/io/ByteArrayOutputStream.h> Inheritance diagram for decaf::io::ByteArrayOutputStream:

Public Member Functions

- **ByteArrayOutputStream** ()
Default Constructor - uses a default internal buffer.
- **ByteArrayOutputStream** (std::vector< unsigned char > &buffer)
Uses the given buffer as the target.
- virtual ~**ByteArrayOutputStream** ()
- virtual void **setBuffer** (std::vector< unsigned char > &buffer)
Sets the internal buffer.
- virtual const unsigned char * **toByteArray** () const
Get a snapshot of the data.
- virtual const std::vector< unsigned char > **toByteArrayRef** () const
Get a snapshot of the data.
- virtual std::size_t **size** () const
Get the Size of the Internal Buffer.
- virtual void **write** (unsigned char c) throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (IOException)
Invokes flush on the target output stream, has no affect.
- virtual void **reset** () throw (IOException)
Clear current Stream contents.
- void **close** () throw (io::IOException)
Invokes close on the target output stream.
- std::string **toString** () const
Converts the bytes in the buffer into a standard C++ string.

- void **writeTo** (**OutputStream** *out) const throw (IOException, lang::exceptions::NullPointerException)
Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count).
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.149.1 Constructor & Destructor Documentation

6.149.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ()

Default Constructor - uses a default internal buffer.

6.149.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (std::vector< unsigned char > & buffer)

Uses the given buffer as the target. Calls setBuffer.

Parameters:

buffer the target buffer.

6.149.1.3 `virtual decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream ()`
[inline, virtual]

6.149.2 Member Function Documentation

6.149.2.1 `void decaf::io::ByteArrayOutputStream::close () throw (io::IOException)`
[inline, virtual]

Invokes close on the target output stream.

Exceptions:

IOException (p. 1820)

Implements `decaf::io::Closeable` (p. 1019).

6.149.2.2 `virtual void decaf::io::ByteArrayOutputStream::flush () throw (IOException)`
[inline, virtual]

Invokes flush on the target output stream, has no affect.

Exceptions:

IOException (p. 1820)

Implements `decaf::io::OutputStream` (p. 2470).

6.149.2.3 `virtual void decaf::io::ByteArrayOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException)`
[inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.149.2.4 `virtual void decaf::io::ByteArrayOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)`
[inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3124).

6.149.2.5 `virtual void decaf::io::ByteArrayOutputStream::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

6.149.2.6 `virtual void decaf::io::ByteArrayOutputStream::reset () throw (
IOException) [virtual]`

Clear current Stream contents.

Exceptions:

IOException (p. 1820)

6.149.2.7 `virtual void decaf::io::ByteArrayOutputStream::setBuffer (std::vector<
unsigned char > & buffer) [virtual]`

Sets the internal buffer. This input stream will wrap around the given buffer and all writes will be performed directly on the buffer. This object does not retain control of the buffer's lifetime however - this is the job of the caller.

Parameters:

buffer The target buffer.

6.149.2.8 `virtual std::size_t decaf::io::ByteArrayOutputStream::size () const
[inline, virtual]`

Get the Size of the Internal Buffer.

Returns:

size of the internal buffer

6.149.2.9 `virtual const unsigned char* decaf::io::ByteArrayOutputStream::toByteArray () const [inline, virtual]`

Get a snapshot of the data.

Returns:

pointer to the data

6.149.2.10 `virtual const std::vector<unsigned char> decaf::io::ByteArrayOutputStream::toByteArrayRef () const [inline, virtual]`

Get a snapshot of the data.

Returns:

reference to the underlying data as a const std::vector<unsigned char>&

6.149.2.11 `std::string decaf::io::ByteArrayOutputStream::toString () const`

Converts the bytes in the buffer into a standard C++ string.

Returns:

a string containing the bytes in the buffer

6.149.2.12 `virtual bool decaf::io::ByteArrayOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3126).

6.149.2.13 `virtual void decaf::io::ByteArrayOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3127).

6.149.2.14 `virtual void decaf::io::ByteArrayOutputStream::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3128).

6.149.2.15 `virtual void decaf::io::ByteArrayOutputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3130).

6.149.2.16 `virtual void decaf::io::ByteArrayOutputStream::wait
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3131).

6.149.2.17 `virtual void decaf::io::ByteArrayOutputStream::write (const unsigned
char * buffer, std::size_t offset, std::size_t len) throw (IOException,
lang::exceptions::NullPointerException) [virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements `decaf::io::OutputStream` (p. 2471).

6.149.2.18 `virtual void decaf::io::ByteArrayOutputStream::write (const
std::vector< unsigned char > & buffer) throw (IOException)
[virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements `decaf::io::OutputStream` (p. 2471).

6.149.2.19 **virtual void decaf::io::ByteArrayOutputStream::write (unsigned char *c*)
throw (IOException)** [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

6.149.2.20 **void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * *out*)
const throw (IOException, lang::exceptions::NullPointerException)**

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count).

The documentation for this class was generated from the following file:

- src/main/decaf/io/ByteArrayOutputStream.h

6.150 decaf::internal::nio::ByteArrayPerspective Class Reference

This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective.

#include <src/main/decaf/internal/nio/ByteArrayPerspective.h>Inheritance diagram for decaf::internal::nio::ByteArrayPerspective:

Public Member Functions

- **ByteArrayPerspective** (std::size_t capacity)
Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayPerspective** (unsigned char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayPerspective** (char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (double *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (float *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (long long *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (int *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (short *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- virtual ~**ByteArrayPerspective** ()
- **ByteArrayPerspective * takeRef** ()
Takes a reference to this Perspective and returns a pointer to this so that a caller can take a ref and get a ref in one call.

- void **returnRef** ()

Returns a reference to this Perspective, when the count is zero it should be deleted.

- int **getReferences** () const

6.150.1 Detailed Description

This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective. Creating an instance of this class implicitly takes a reference to it, so a creator must return its ref before the count will be zero.

6.150.2 Constructor & Destructor Documentation

6.150.2.1 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (std::size_t capacity)

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.150.2.2 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (unsigned char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.3 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.4 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (double * *array*, std::size_t *capacity*, bool *own* = false) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.5 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (float * *array*, std::size_t *capacity*, bool *own* = false) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.6 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (long long * *array*, std::size_t *capacity*, bool *own* = false) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.7 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (int * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.8 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (short * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.150.2.9 virtual decaf::internal::nio::ByteArrayPerspective::~~ByteArrayPerspective () [inline, virtual]

6.150.3 Member Function Documentation

6.150.3.1 int decaf::internal::nio::ByteArrayPerspective::getReferences () const [inline]

Returns:

the current number of reference on this perspective

6.150.3.2 `void decaf::internal::nio::ByteArrayPerspective::returnRef () [inline]`

Returns a reference to this Perspective, when the count is zero it should be deleted.

6.150.3.3 `ByteArrayPerspective* decaf::internal::nio::ByteArrayPerspective::takeRef () [inline]`

Takes a reference to this Perspective and returns a pointer to this so that a caller can take a ref and get a ref in one call.

Returns:

this

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteArrayPerspective.h`

6.151 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:.

```
#include <src/main/decaf/nio/ByteBuffer.h> Inheritance diagram for decaf::nio::ByteBuffer:
```

Public Member Functions

- virtual **~ByteBuffer** ()
- virtual std::string **toString** () const
- **ByteBuffer** & **get** (std::vector< unsigned char > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **ByteBuffer** & **get** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- **ByteBuffer** & **put** (**ByteBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the bytes remaining in the given source buffer into this buffer.
- **ByteBuffer** & **put** (const unsigned char *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers bytes into this buffer from the given source array.
- **ByteBuffer** & **put** (std::vector< unsigned char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source byte array into this buffer.
- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.
- virtual unsigned char * **array** ()=0 throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)
Returns the byte array that backs this buffer.
- virtual std::size_t **arrayOffset** () const =0 throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)
Returns the offset within this buffer's backing array of the first element of the buffer.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible byte array.
- virtual **CharBuffer** * **asCharBuffer** () const =0
Creates a view of this byte buffer as a char buffer.

- virtual **DoubleBuffer** * **asDoubleBuffer** () const =0
Creates a view of this byte buffer as a double buffer.
- virtual **FloatBuffer** * **asFloatBuffer** () const =0
Creates a view of this byte buffer as a float buffer.
- virtual **IntBuffer** * **asIntBuffer** () const =0
Creates a view of this byte buffer as a int buffer.
- virtual **LongBuffer** * **asLongBuffer** () const =0
Creates a view of this byte buffer as a long buffer.
- virtual **ShortBuffer** * **asShortBuffer** () const =0
Creates a view of this byte buffer as a short buffer.
- virtual **ByteBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only byte buffer that shares this buffer's content.
- virtual **ByteBuffer** & **compact** ()=0 throw (**ReadOnlyBufferException**)
Compacts this buffer.
- virtual **ByteBuffer** * **duplicate** ()=0
Creates a new byte buffer that shares this buffer's content.
- virtual unsigned char **get** () const =0 throw (**BufferUnderflowException**)
Relative get method.
- virtual unsigned char **get** (std::size_t index) const =0 throw (**lang::exceptions::IndexOutOfBoundsException**)
Absolute get method.
- virtual char **getChar** ()=0 throw (**BufferUnderflowException**)
Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual char **getChar** (std::size_t index) const =0 throw (**lang::exceptions::IndexOutOfBoundsException**)
Reads one byte at the given index and returns it.
- virtual double **getDouble** ()=0 throw (**BufferUnderflowException**)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (std::size_t index) const =0 throw (**lang::exceptions::IndexOutOfBoundsException**)
Reads eight bytes at the given index and returns it.
- virtual float **getFloat** ()=0 throw (**BufferUnderflowException**)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual float **getFloat** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual long long **getLong** ()=0 throw (BufferUnderflowException)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual long long **getLong** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual int **getInt** ()=0 throw (BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual int **getInt** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual short **getShort** ()=0 throw (BufferUnderflowException)
Reads the next two bytes at this buffer's current position, and then increments the position by that amount.
- virtual short **getShort** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given index and returns it.
- virtual **ByteBuffer** & **put** (unsigned char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given byte into this buffer at the current position, and then increments the position.
- virtual **ByteBuffer** & **put** (std::size_t index, unsigned char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given byte into this buffer at the given index.
- virtual **ByteBuffer** & **putChar** (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.
- virtual **ByteBuffer** & **putChar** (std::size_t index, char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putDouble** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putDouble** (std::size_t index, double value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putFloat** (float value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putFloat** (std::size_t index, float value)=0 throw (lang::exceptions::IndexOutOfBoundsException, **ReadOnlyBufferException**)

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putLong** (long long value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putLong** (std::size_t index, long long value)=0 throw (lang::exceptions::IndexOutOfBoundsException, **ReadOnlyBufferException**)

Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putInt** (int value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putInt** (std::size_t index, int value)=0 throw (lang::exceptions::IndexOutOfBoundsException, **ReadOnlyBufferException**)

Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putShort** (short value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putShort** (std::size_t index, short value)=0 throw (lang::exceptions::IndexOutOfBoundsException, **ReadOnlyBufferException**)

Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** * **slice** () const =0

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

- virtual int **compareTo** (const **ByteBuffer** &value) const

Compares this object with the specified object for order.

- virtual bool **equals** (const **ByteBuffer** &value) const

- virtual bool **operator==** (const **ByteBuffer** &value) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const **ByteBuffer** &value) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **ByteBuffer** * **allocate** (std::size_t capacity)
Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **ByteBuffer** * **wrap** (unsigned char *array, std::size_t offset, std::size_t length)
 throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **ByteBuffer** (p. 913).*
- static **ByteBuffer** * **wrap** (std::vector< unsigned char > &buffer)
*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 913).*

Protected Member Functions

- **ByteBuffer** (std::size_t capacity)
*Creates a **ByteBuffer** (p. 913) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.151.1 Detailed Description

This class defines six categories of operations upon byte buffers:. 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 926) float getFloat(int index) void **putFloat(float f)** (p. 933) void putFloat(int index, float f)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in

the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The `asFloatBuffer` method, for example, creates an instance of the **FloatBuffer** (p.1665) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types `char`, `short`, `int`, `long`, and `double`.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.151.2 Constructor & Destructor Documentation

6.151.2.1 `decaf::nio::ByteBuffer::ByteBuffer (std::size_t capacity)` [protected]

Creates a **ByteBuffer** (p.913) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.151.2.2 `virtual decaf::nio::ByteBuffer::~~ByteBuffer ()` [inline, virtual]

6.151.3 Member Function Documentation

6.151.3.1 `static ByteBuffer* decaf::nio::ByteBuffer::allocate (std::size_t capacity)` [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the internal buffer's capacity.

Returns:

a newly allocated **ByteBuffer** (p.913) which the caller owns.

6.151.3.2 `virtual unsigned char* decaf::nio::ByteBuffer::array () throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)` [pure virtual]

Returns the byte array that backs this buffer. Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The array that backs this buffer

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteBuffer** (p. 876).

6.151.3.3 `virtual std::size_t decaf::nio::ByteBuffer::arrayOffset
() const throw (ReadOnlyBufferException,
lang::exceptions::UnsupportedOperationException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer. If this buffer is backed by an array then buffer position *p* corresponds to array index *p* + **arrayOffset()** (p. 919).

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset within this buffer's array of the first element of the buffer

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteBuffer** (p. 876).

6.151.3.4 `virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer () const [pure
virtual]`

Creates a view of this byte buffer as a char buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new **Char Buffer** (p. 803), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 876).

6.151.3.5 `virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer () const
[pure virtual]`

Creates a view of this byte buffer as a double buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new double **Buffer** (p. 803), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 877).

6.151.3.6 virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer () const [pure virtual]

Creates a view of this byte buffer as a float buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new float **Buffer** (p. 803), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 877).

6.151.3.7 virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer () const [pure virtual]

Creates a view of this byte buffer as a int buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new int **Buffer** (p. 803), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 877).

6.151.3.8 virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer () const [pure virtual]

Creates a view of this byte buffer as a long buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new long **Buffer** (p. 803), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 878).

6.151.3.9 virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer () const
[pure virtual]

Creates a new, read-only byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 878).

6.151.3.10 virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer () const
[pure virtual]

Creates a view of this byte buffer as a short buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new short **Buffer** (p. 803), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 878).

6.151.3.11 virtual ByteBuffer& decaf::nio::ByteBuffer::compact () throw (
ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n + 1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **ByteBuffer** (p. 913)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 879).

6.151.3.12 **virtual int decaf::nio::ByteBuffer::compareTo (const ByteBuffer & value) const** [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.151.3.13 **virtual ByteBuffer* decaf::nio::ByteBuffer::duplicate ()** [pure virtual]

Creates a new byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new Byte **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 879).

6.151.3.14 **virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & value) const** [virtual]

Returns:

true if this value is considered equal to the passed value.

6.151.3.15 **virtual unsigned char decaf::nio::ByteBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)** [pure virtual]

Absolute get method. Reads the byte at the given index.

Parameters:

index - the index in the **Buffer** (p. 803) where the byte is to be read

Returns:

the byte that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ByteBuffer** (p. 879).

6.151.3.16 virtual unsigned char decaf::nio::ByteBuffer::get () const throw (**BufferUnderflowException**) [pure virtual]

Relative get method. Reads the byte at this buffer's current position, and then increments the position.

Returns:

The byte at the buffer's current position

Exceptions:

BufferUnderflowException (p. 837) - If the buffer's current position is not smaller than its limit

Implemented in **decaf::internal::nio::ByteBuffer** (p. 880).

6.151.3.17 ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (**BufferUnderflowException**, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies `length` bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` bytes remaining in this buffer

NullPointerException if the passed buffer is null.

6.151.3.18 `ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char > buffer) throw (BufferUnderflowException)`

Relative bulk get method. This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns:

a reference to this Byte **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than length bytes remaining in this buffer

6.151.3.19 `virtual char decaf::nio::ByteBuffer::getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Reads one byte at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 803) where the byte is to be read

Returns:

the char at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 880).

6.151.3.20 `virtual char decaf::nio::ByteBuffer::getChar () throw (BufferUnderflowException) [pure virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns:

the next char in the buffer..

Exceptions:

BufferUnderflowException (p. 837) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 880).

6.151.3.21 `virtual double decaf::nio::ByteBuffer::getDouble (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 803) where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 881).

6.151.3.22 `virtual double decaf::nio::ByteBuffer::getDouble () throw (
BufferUnderflowException) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next double in the buffer..

Exceptions:

BufferUnderflowException (p. 837) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 881).

6.151.3.23 `virtual float decaf::nio::ByteBuffer::getFloat (std::size_t index) const
throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 803) where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 881).

6.151.3.24 **virtual float decaf::nio::ByteBuffer::getFloat () throw (**
 BufferUnderflowException) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next float in the buffer..

Exceptions:

BufferUnderflowException (p. 837) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 882).

6.151.3.25 **virtual int decaf::nio::ByteBuffer::getInt (std::size_t index) const throw**
 (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 803) where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 882).

6.151.3.26 **virtual int decaf::nio::ByteBuffer::getInt () throw (**
 BufferUnderflowException) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next int in the buffer..

Exceptions:

BufferUnderflowException (p. 837) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 883).

6.151.3.27 `virtual long long decaf::nio::ByteBuffer::getLong (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 803) where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 883).

6.151.3.28 `virtual long long decaf::nio::ByteBuffer::getLong () throw (
BufferUnderflowException) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next long long in the buffer..

Exceptions:

BufferUnderflowException (p. 837) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 883).

6.151.3.29 `virtual short decaf::nio::ByteBuffer::getShort (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads two bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 803) where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 884).

6.151.3.30 virtual short decaf::nio::ByteBuffer::getShort () throw (BufferUnderflowException) [pure virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next short in the buffer..

Exceptions:

BufferUnderflowException (p. 837) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 884).

6.151.3.31 virtual bool decaf::nio::ByteBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible byte array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 884).

6.151.3.32 virtual bool decaf::nio::ByteBuffer::isReadOnly () const [pure virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 806).

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 884).

6.151.3.33 virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer & value) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.151.3.34 virtual bool decaf::nio::ByteBuffer::operator==(const ByteBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.151.3.35 virtual ByteBuffer& decaf::nio::ByteBuffer::put (std::size_t *index*, unsigned char *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given byte into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data
value - the byte to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 885).

6.151.3.36 virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters:

value - the byte value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 885).

6.151.3.37 `ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)`

This method transfers the entire content of the given source byte array into this buffer. This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters:

buffer - The buffer whose contents are copied to this `ByteBuffer` (p. 913)

Returns:

a reference to this buffer

Exceptions:

`BufferOverflowException` (p. 834) - If there is insufficient space in this buffer

`ReadOnlyBufferException` (p. 2685) - If this buffer is read-only

6.151.3.38 `ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)`

This method transfers bytes into this buffer from the given source array. If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a `BufferOverflowException` (p. 834) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters:

buffer - The array from which bytes are to be read

offset - The offset within the array of the first byte to be read;

length - The number of bytes to be read from the given array

Returns:

a reference to this buffer

Exceptions:

`BufferOverflowException` (p. 834) - If there is insufficient space in this buffer

`ReadOnlyBufferException` (p. 2685) - If this buffer is read-only

`NullPointerException` if the passed buffer is null.

6.151.3.39 `ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)`

This method transfers the bytes remaining in the given source buffer into this buffer. If there are more bytes remaining in the source buffer than in this buffer, that is, if `src.remaining() >`

remaining() (p. 808), then no bytes are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies $n = \text{src.remaining}()$ bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n .

Parameters:

src - the buffer to take bytes from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining bytes in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.151.3.40 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes one byte containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 886).

6.151.3.41 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 886).

6.151.3.42 `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 886).

6.151.3.43 `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 887).

6.151.3.44 `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data
value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 887).

6.151.3.45 `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 888).

6.151.3.46 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 888).

6.151.3.47 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 889).

6.151.3.48 `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 889).

6.151.3.49 `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long value)
 throw (BufferOverflowException, ReadOnlyBufferException)` [pure
 virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 889).

6.151.3.50 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (std::size_t index,
 short value) throw (lang::exceptions::IndexOutOfBoundsException,
 ReadOnlyBufferException)` [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteBuffer` (p. 890).

6.151.3.51 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (short value)
 throw (BufferOverflowException, ReadOnlyBufferException)` [pure
 virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 890).

6.151.3.52 virtual ByteBuffer* decaf::nio::ByteBuffer::slice () const [pure virtual]

Creates a new byte buffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ByteBuffer** (p.913) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 891).

6.151.3.53 virtual std::string decaf::nio::ByteBuffer::toString () const [virtual]**Returns:**

a std::string describing this object

6.151.3.54 static ByteBuffer* decaf::nio::ByteBuffer::wrap (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a **ByteBuffer** (p.913). The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new **ByteBuffer** (p.913) that is backed by *buffer*, caller owns.

6.151.3.55 `static ByteBuffer* decaf::nio::ByteBuffer::wrap (unsigned
char * array, std::size_t offset, std::size_t length) throw (
lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **ByteBuffer** (p. 913). The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

- array* - The array that will back the new buffer
- offset* - The offset of the subarray to be used
- length* - The length of the subarray to be used

Returns:

- a new **ByteBuffer** (p. 913) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ByteBuffer.h`

6.152 cms::BytesMessage Class Reference

A **BytesMessage** (p. 938) object is used to send a message containing a stream of unsigned bytes.

#include <src/main/cms/BytesMessage.h> Inheritance diagram for cms::BytesMessage:

Public Member Functions

- virtual **~BytesMessage** ()
- virtual void **setBodyBytes** (const unsigned char *buffer, std::size_t numBytes)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const =0 throw (cms::MessageNotReadableException, cms::CMSEException)
Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.
- virtual std::size_t **getBodyLength** () const =0 throw (cms::MessageNotReadableException, cms::CMSEException)
Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()=0 throw (cms::MessageFormatException, cms::CMSEException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the bytes message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

- virtual `std::size_t readBytes` (`unsigned char *buffer`, `std::size_t length`) `const =0` throw (`cms::MessageEOFException`, `cms::MessageNotReadableException`, `cms::CMSEException`)
Reads a portion of the bytes message stream.
- virtual void `writeBytes` (`const unsigned char *value`, `std::size_t offset`, `std::size_t length`)=`0` throw (`cms::MessageNotWriteableException`, `cms::CMSEException`)
Writes a portion of a byte array to the bytes message stream.
- virtual `char readChar` () `const =0` throw (`cms::MessageEOFException`, `cms::MessageNotReadableException`, `cms::CMSEException`)
Reads a Char from the Bytes message stream.
- virtual void `writeChar` (`char value`)=`0` throw (`cms::MessageNotWriteableException`, `cms::CMSEException`)
Writes a char to the bytes message stream as a 1-byte value.
- virtual `float readFloat` () `const =0` throw (`cms::MessageEOFException`, `cms::MessageNotReadableException`, `cms::CMSEException`)
Reads a 32 bit float from the Bytes message stream.
- virtual void `writeFloat` (`float value`)=`0` throw (`cms::MessageNotWriteableException`, `cms::CMSEException`)
Writes a float to the bytes message stream as a 4 byte value.
- virtual `double readDouble` () `const =0` throw (`cms::MessageEOFException`, `cms::MessageNotReadableException`, `cms::CMSEException`)
Reads a 64 bit double from the Bytes message stream.
- virtual void `writeDouble` (`double value`)=`0` throw (`cms::MessageNotWriteableException`, `cms::CMSEException`)
Writes a double to the bytes message stream as a 8 byte value.
- virtual `short readShort` () `const =0` throw (`cms::MessageEOFException`, `cms::MessageNotReadableException`, `cms::CMSEException`)
Reads a 16 bit signed short from the Bytes message stream.
- virtual void `writeShort` (`short value`)=`0` throw (`cms::MessageNotWriteableException`, `cms::CMSEException`)
Writes a signed short to the bytes message stream as a 2 byte value.
- virtual `unsigned short readUnsignedShort` () `const =0` throw (`cms::MessageEOFException`, `cms::MessageNotReadableException`, `cms::CMSEException`)
Reads a 16 bit unsigned short from the Bytes message stream.
- virtual void `writeUnsignedShort` (`unsigned short value`)=`0` throw (`cms::MessageNotWriteableException`, `cms::CMSEException`)
Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit signed integer from the Bytes message stream.
- virtual void **writeInt** (int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed int to the bytes message stream as a 4 byte value.
- virtual long long **readLong** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit long from the Bytes message stream.
- virtual void **writeLong** (long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a long long to the bytes message stream as a 8 byte value.
- virtual std::string **readString** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads an ASCII String from the Bytes message stream.
- virtual void **writeString** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an ASCII String to the Bytes message stream.
- virtual std::string **readUTF** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
*Reads an UTF String from the **BytesMessage** (p. 938) stream.*
- virtual void **writeUTF** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
*Writes an UTF String to the **BytesMessage** (p. 938) stream.*
- virtual **BytesMessage** * **clone** () const =0
Clones this message.

6.152.1 Detailed Description

A **BytesMessage** (p. 938) object is used to send a message containing a stream of unsigned bytes. It inherits from the **Message** (p. 2163) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 938) interface.

The **BytesMessage** (p. 938) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1379) and **decaf.io.DataOutputStream** (p. 1388).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when `clearBody` is called, the body of the message is in write-only mode. After the first call to `reset` has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called `reset` so that the message body is in read-only mode for the client.

If `clearBody` is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 2330) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 2331) is thrown.

Since:

1.0

6.152.2 Constructor & Destructor Documentation

6.152.2.1 `virtual cms::BytesMessage::~~BytesMessage () [inline, virtual]`

6.152.3 Member Function Documentation

6.152.3.1 `virtual BytesMessage* cms::BytesMessage::clone () const [pure virtual]`

Clones this message.

Returns:

a deep copy of this message.

Exceptions:

CMSException (p. 1031) - if an internal error occurs while cloning the **Message** (p. 2163).

Implements **cms::Message** (p. 2168).

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 200).

6.152.3.2 `virtual unsigned char* cms::BytesMessage::getBodyBytes () const throw (cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller. This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

Returns:

pointer to a byte buffer that the call owns upon completion of this method.

Exceptions:

CMSException (p. 1031) - If an internal error occurs.

MessageNotReadableException (p. 2330) - If the message is in Write Only Mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 201).

6.152.3.3 `virtual std::size_t cms::BytesMessage::getBodyLength () const throw (cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Returns the number of bytes contained in the body of this message.

Returns:

number of bytes.

Exceptions:

CMSException (p. 1031) - If an internal error occurs.

MessageNotReadableException (p. 2330) - If the message is in Write Only Mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 202).

6.152.3.4 `virtual bool cms::BytesMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Boolean from the Bytes message stream.

Returns:

boolean value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 202).

6.152.3.5 `virtual unsigned char cms::BytesMessage::readByte () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Byte from the Bytes message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 203).

6.152.3.6 `virtual std::size_t cms::BytesMessage::readBytes (unsigned char *
buffer, std::size_t length) const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [pure
virtual]`

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 203).

6.152.3.7 `virtual std::size_t cms::BytesMessage::readBytes (std::vector<
unsigned char > &value) const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [pure
virtual]`

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 204).

6.152.3.8 `virtual char cms::BytesMessage::readChar () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException) [pure virtual]`

Reads a Char from the Bytes message stream.

Returns:

char value from stream

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 204).

6.152.3.9 `virtual double cms::BytesMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException) [pure virtual]`

Reads a 64 bit double from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 204).

6.152.3.10 `virtual float cms::BytesMessage::readFloat () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 32 bit float from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 205).

6.152.3.11 `virtual int cms::BytesMessage::readInt () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

Returns:

int value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 205).

6.152.3.12 `virtual long long cms::BytesMessage::readLong () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit long from the Bytes message stream.

Returns:

long long value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 205).

6.152.3.13 `virtual short cms::BytesMessage::readShort () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 16 bit signed short from the Bytes message stream.

Returns:

short value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 206).

6.152.3.14 `virtual std::string cms::BytesMessage::readString () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads an ASCII String from the Bytes message stream.

Returns:

String from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 206).

6.152.3.15 `virtual unsigned short cms::BytesMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 16 bit unsigned short from the Bytes message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.
MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 207).

6.152.3.16 `virtual std::string cms::BytesMessage::readUTF () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads an UTF String from the **BytesMessage** (p. 938) stream.

Returns:

String from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.
MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.
MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 207).

6.152.3.17 `virtual void cms::BytesMessage::reset () throw (cms::MessageFormatException, cms::CMSException) [pure virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions:

CMSException (p. 1031) - If the provider fails to perform the reset operation.
MessageFormatException (p. 2278) - If the **Message** (p. 2163) has an invalid format.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 207).

6.152.3.18 `virtual void cms::BytesMessage::setBodyBytes (const unsigned char * buffer, std::size_t numBytes) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

sets the bytes given to the message body.

Parameters:

buffer Byte Buffer to copy
numBytes Number of bytes in Buffer to copy

Exceptions:

CMSException (p. 1031) - If an internal error occurs.
MessageNotWriteableException (p. 2331) - if in Read Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 208).

6.152.3.19 `virtual void cms::BytesMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value boolean to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 208).

6.152.3.20 `virtual void cms::BytesMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters:

value byte to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 208).

6.152.3.21 `virtual void cms::BytesMessage::writeBytes (const unsigned char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

Parameters:

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 209).

6.152.3.22 `virtual void cms::BytesMessage::writeBytes (const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters:

value bytes to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 209).

6.152.3.23 `virtual void cms::BytesMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a char to the bytes message stream as a 1-byte value.

Parameters:

value char to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 209).

6.152.3.24 `virtual void cms::BytesMessage::writeDouble (double value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a double to the bytes message stream as a 8 byte value.

Parameters:

value double to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 210).

6.152.3.25 `virtual void cms::BytesMessage::writeFloat (float value) throw (cms::MessageNotWriteableException, cms::CMSException)` [pure virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters:

value float to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 210).

6.152.3.26 `virtual void cms::BytesMessage::writeInt (int value) throw (cms::MessageNotWriteableException, cms::CMSException)` [pure virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters:

value signed int to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 210).

6.152.3.27 `virtual void cms::BytesMessage::writeLong (long long value) throw (cms::MessageNotWriteableException, cms::CMSException)` [pure virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters:

value signed long long to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 211).

6.152.3.28 `virtual void cms::BytesMessage::writeShort (short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters:

value signed short to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 211).

6.152.3.29 `virtual void cms::BytesMessage::writeString (const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes an ASCII String to the Bytes message stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 211).

6.152.3.30 `virtual void cms::BytesMessage::writeUnsignedShort (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value unsigned short to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 212).

6.152.3.31 `virtual void cms::BytesMessage::writeUTF (const std::string & value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[pure virtual]`

Writes an UTF String to the `BytesMessage` (p. 938) stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 212).

The documentation for this class was generated from the following file:

- `src/main/cms/BytesMessage.h`

6.153 activemq::cmsutil::CachedConsumer Class Reference

A cached message consumer contained within a pooled session.

#include <src/main/activemq/cmsutil/CachedConsumer.h> Inheritance diagram for activemq::cmsutil::CachedConsumer:

Public Member Functions

- **CachedConsumer** (cms::MessageConsumer *consumer)
- virtual ~**CachedConsumer** ()
- virtual void **close** () throw (cms::CMSEException)
Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual cms::Message * **receive** () throw (cms::CMSEException)
Synchronously Receive a Message.
- virtual cms::Message * **receive** (int millisecs) throw (cms::CMSEException)
Synchronously Receive a Message, time out after defined interval.
- virtual cms::Message * **receiveNoWait** () throw (cms::CMSEException)
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (cms::MessageListener *listener) throw (cms::CMSEException)
Sets the MessageListener that this class will send notifs on.
- virtual cms::MessageListener * **getMessageListener** () const throw (cms::CMSEException)
Gets the MessageListener that this class will send new Message notification events to.
- virtual std::string **getMessageSelector** () const throw (cms::CMSEException)
Gets this message consumer's message selector expression.

6.153.1 Detailed Description

A cached message consumer contained within a pooled session.

6.153.2 Constructor & Destructor Documentation

6.153.2.1 `activemq::cmsutil::CachedConsumer::CachedConsumer (cms::MessageConsumer * consumer) [inline]`

6.153.2.2 `virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer () [inline, virtual]`

6.153.3 Member Function Documentation

6.153.3.1 `virtual void activemq::cmsutil::CachedConsumer::close () throw (cms::CMSEException) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p. 1021).

6.153.3.2 `virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener () const throw (cms::CMSEException) [inline, virtual]`

Gets the MessageListener that this class will send new Message notification events to.

Returns:

The listener of messages received by this consumer

Exceptions:

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2215).

6.153.3.3 `virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector () const throw (cms::CMSEException) [inline, virtual]`

Gets this message consumer's message selector expression.

Returns:

This Consumer's selector expression or "".

Exceptions:

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2215).

6.153.3.4 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive (int millisecs) throw (cms::CMSEException) [inline, virtual]`

Synchronously Receive a Message, time out after defined interval. Returns null if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2215).

6.153.3.5 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive ()
throw (cms::CMSEException) [inline, virtual]`

Synchronously Receive a Message.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2216).

6.153.3.6 `virtual cms::Message* ac-
tivemq::cmsutil::CachedConsumer::receiveNoWait ()
throw (cms::CMSEException) [inline, virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2216).

6.153.3.7 `virtual void activemq::cmsutil::CachedConsumer::setMessageListener
(cms::MessageListener * listener) throw (cms::CMSEException)
[inline, virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters:

listener The listener of messages received by this consumer.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2216).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedConsumer.h`

6.154 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

#include <src/main/activemq/cmsutil/CachedProducer.h> Inheritance diagram for activemq::cmsutil::CachedProducer:

Public Member Functions

- **CachedProducer** (**cms::MessageProducer** *producer)
- virtual **~CachedProducer** ()
- virtual void **close** () throw (cms::CMSEException)
Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual void **send** (**cms::Message** *message) throw (cms::CMSEException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (**cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message) throw (cms::CMSEException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **setDeliveryMode** (int mode) throw (cms::CMSEException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const throw (cms::CMSEException)
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value) throw (cms::CMSEException)
Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () const throw (cms::CMSEException)
Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSEException)
Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSEException)

Gets if Message Time Stamps are disabled for this Producer.

- virtual void **setPriority** (int priority) throw (cms::CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const throw (cms::CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time) throw (cms::CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const throw (cms::CMSEException)
Gets the Time to Live that this producer sends messages with.

6.154.1 Detailed Description

A cached message producer contained within a pooled session.

6.154.2 Constructor & Destructor Documentation

6.154.2.1 `activemq::cmsutil::CachedProducer::CachedProducer (cms::MessageProducer * producer) [inline]`

6.154.2.2 `virtual activemq::cmsutil::CachedProducer::~~CachedProducer () [inline, virtual]`

6.154.3 Member Function Documentation

6.154.3.1 `virtual void activemq::cmsutil::CachedProducer::close () throw (cms::CMSEException) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p.1021).

6.154.3.2 `virtual int activemq::cmsutil::CachedProducer::getDeliveryMode () const throw (cms::CMSEException) [inline, virtual]`

Gets the delivery mode for this Producer.

Returns:

The DeliveryMode

Exceptions:

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p.2333).

6.154.3.3 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID () const throw (cms::CMSEException) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2334).

6.154.3.4 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp () const throw (cms::CMSEException) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2334).

6.154.3.5 `virtual int activemq::cmsutil::CachedProducer::getPriority () const throw (cms::CMSEException) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns:

int based priority level

Exceptions:

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2334).

6.154.3.6 `virtual long long activemq::cmsutil::CachedProducer::getTimeToLive () const throw (cms::CMSEException) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns:

Time to live value in milliseconds

Exceptions:

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2335).

6.154.3.7 `virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)` [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2335).

6.154.3.8 `virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * destination, cms::Message * message) throw (cms::CMSEException)` [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

destination The destination on which to send the message

message the message to be sent.

Exceptions:

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2335).

6.154.3.9 **virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException)** [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2336).

6.154.3.10 **virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message) throw (cms::CMSException)** [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

message The message to be sent.

Exceptions:

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2336).

6.154.3.11 `virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int mode) throw (cms::CMSException) [inline, virtual]`

Sets the delivery mode for this Producer.

Parameters:

mode The DeliveryMode

Exceptions:

CMSException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2337).

6.154.3.12 `virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool value) throw (cms::CMSException) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters:

value boolean indicating enable / disable (true / false)

Exceptions:

CMSException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2337).

6.154.3.13 `virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool value) throw (cms::CMSException) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Exceptions:

CMSException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2337).

6.154.3.14 `virtual void activemq::cmsutil::CachedProducer::setPriority (int priority) throw (cms::CMSException) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters:

priority int value for Priority level

Exceptions:

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2338).

6.154.3.15 `virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long long time) throw (cms::CMSEException) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with. This value will be used if the time to live is not specified via the send method.

Parameters:

time default time to live value in milliseconds

Exceptions:

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2338).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedProducer.h`

6.155 decaf::util::concurrent::Callable< V > Class Template Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

Public Member Functions

- virtual `~Callable ()`
- virtual `V call ()=0 throw (decaf::lang::Exception)`
Computes a result, or throws an exception if unable to do so.

6.155.1 Detailed Description

```
template<typename V> class decaf::util::concurrent::Callable< V >
```

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called `call`. This interface differs from the `Runnable` interface in that a **Callable** (p. 964) object can return a result and is allowed to throw an exceptions from its `call` method.

The `Executors` class contains utility methods to convert from other common forms to **Callable** (p. 964) classes.

Since:

1.0

6.155.2 Constructor & Destructor Documentation

6.155.2.1 `template<typename V > virtual decaf::util::concurrent::Callable< V >::~~Callable () [inline, virtual]`

6.155.3 Member Function Documentation

6.155.3.1 `template<typename V > virtual V decaf::util::concurrent::Callable< V >::call () throw (decaf::lang::Exception) [pure virtual]`

Computes a result, or throws an exception if unable to do so.

Returns:

Computed Result.

Exceptions:

Exception If unable to compute a result.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Callable.h`

6.156 decaf::util::concurrent::CancellationException Class Reference

#include <src/main/decaf/util/concurrent/CancellationException.h> Inheritance diagram for decaf::util::concurrent::CancellationException:

Public Member Functions

- **CancellationException** () throw ()
Default Constructor.
- **CancellationException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CancellationException** (const CancellationException &ex) throw ()
Copy Constructor.
- **CancellationException** (const std::exception *cause) throw ()
Constructor.
- **CancellationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **CancellationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CancellationException** * clone () const
Clones this exception.
- virtual ~**CancellationException** () throw ()

6.156.1 Constructor & Destructor Documentation

6.156.1.1 decaf::util::concurrent::CancellationException::CancellationException () throw () [inline]

Default Constructor.

6.156.1.2 decaf::util::concurrent::CancellationException::CancellationException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.156.1.3 `decaf::util::concurrent::CancellationException::CancellationException` (`const CancellationException & ex`) `throw ()` [inline]

Copy Constructor.

Parameters:

ex - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

6.156.1.4 `decaf::util::concurrent::CancellationException::CancellationException` (`const std::exception * cause`) `throw ()` [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.156.1.5 `decaf::util::concurrent::CancellationException::CancellationException` (`const char * file`, `const int lineNumber`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.156.1.6 `decaf::util::concurrent::CancellationException::CancellationException` (`const char * file`, `const int lineNumber`, `const std::exception * cause`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.156.1.7 **virtual**
 decaf::util::concurrent::CancellationException::~~CancellationException ()
 throw () [inline, virtual]

6.156.2 Member Function Documentation

6.156.2.1 **virtual CancellationException* de-**
 caf::util::concurrent::CancellationException::clone () const
 [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CancellationException.h**

6.157 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

#include <src/main/decaf/security/cert/Certificate.h> Inheritance diagram for decaf::security::cert::Certificate:

Public Member Functions

- virtual **~Certificate** ()
- virtual bool **equals** (const **Certificate** &cert) const =0
Compares the encoded form of the two certificates.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
Provides the encoded form of this certificate.
- virtual std::string **getType** () const =0
Returns the type of this certificate.
- virtual **PublicKey** * **getPublicKey** ()=0
Gets the public key of this certificate.
- virtual const **PublicKey** * **getPublicKey** () const =0
Gets the public key of this certificate.
- virtual void **verify** (const **PublicKey** &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual std::string **toString** () const =0
Returns a string representation of this certificate.

6.157.1 Detailed Description

Base interface for all identity certificates.

6.157.2 Constructor & Destructor Documentation

6.157.2.1 `virtual decaf::security::cert::Certificate::~~Certificate () [inline, virtual]`

6.157.3 Member Function Documentation

6.157.3.1 `virtual bool decaf::security::cert::Certificate::equals (const Certificate & cert) const [pure virtual]`

Compares the encoded form of the two certificates.

Parameters:

cert The certificate to be tested for equality with this certificate.

Returns:

true if the given certificate is equal to this certificate.

6.157.3.2 `virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & output) const throw (CertificateEncodingException) [pure virtual]`

Provides the encoded form of this certificate.

Parameters:

output Receives the encoded form of this certificate.

Exceptions:

CertificateEncodingException (p. 972) if an encoding error occurs

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2444).

6.157.3.3 `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey () const [pure virtual]`

Gets the public key of this certificate.

Returns:

the public key

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2445).

6.157.3.4 `virtual PublicKey* decaf::security::cert::Certificate::getPublicKey () [pure virtual]`

Gets the public key of this certificate.

Returns:

the public key

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2445).

6.157.3.5 `virtual std::string decaf::security::cert::Certificate::getType () const`
[pure virtual]

Returns the type of this certificate.

Returns:

the type of this certificate

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2446).

6.157.3.6 `virtual std::string decaf::security::cert::Certificate::toString () const`
[pure virtual]

Returns a string representation of this certificate.

Returns:

a string representation of this certificate

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2446).

6.157.3.7 `virtual void decaf::security::cert::Certificate::verify (const
PublicKey & publicKey, const std::string & sigProvider) const
throw (NoSuchAlgorithmException, InvalidKeyException,
NoSuchProviderException, SignatureException, CertificateException)`
[pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key. Uses the verification engine of the specified provider.

Parameters:

publicKey The public key used to carry out the validation.

sigProvider The name of the signature provider

Exceptions:

NoSuchAlgorithmException (p. 2420) - on unsupported signature algorithms.

InvalidKeyException (p. 1810) - on incorrect key.

NoSuchProviderException (p. 2426) - if there's no default provider.

SignatureException (p. 2957) - on signature errors.

CertificateException (p. 974) - on encoding errors.

6.157.3.8 virtual void decaf::security::cert::Certificate::verify (const PublicKey & *publicKey*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters:

publicKey The public key used to carry out the validation.

Exceptions:

NoSuchAlgorithmException (p. 2420) - on unsupported signature algorithms.

InvalidKeyException (p. 1810) - on incorrect key.

NoSuchProviderException (p. 2426) - if there's no default provider.

SignatureException (p. 2957) - on signature errors.

CertificateException (p. 974) - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

6.158 decaf::security::cert::CertificateEncodingException Class Reference

#include <src/main/decaf/security/cert/CertificateEncodingException.h> Inheritance diagram for decaf::security::cert::CertificateEncodingException:

Public Member Functions

- **CertificateEncodingException** () throw ()
Default Constructor.
- **CertificateEncodingException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()
Copy Constructor.
- **CertificateEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateEncodingException** * clone () const
Clones this exception.
- virtual ~**CertificateEncodingException** () throw ()

6.158.1 Constructor & Destructor Documentation

6.158.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException () throw () [inline]

Default Constructor.

6.158.1.2 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.158.1.3 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const CertificateEncodingException & ex) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.158.1.4 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.158.1.5 `virtual decaf::security::cert::CertificateEncodingException::~CertificateEncodingException () throw () [inline, virtual]`

6.158.2 Member Function Documentation

6.158.2.1 `virtual CertificateEncodingException* decaf::security::cert::CertificateEncodingException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.975).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateEncodingException.h`

6.159 decaf::security::cert::CertificateException Class Reference

#include <src/main/decaf/security/cert/CertificateException.h> Inheritance diagram for decaf::security::cert::CertificateException:

Public Member Functions

- **CertificateException** () throw ()
Default Constructor.
- **CertificateException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateException** (const **CertificateException** &ex) throw ()
Copy Constructor.
- **CertificateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateException** * **clone** () const
Clones this exception.
- virtual ~**CertificateException** () throw ()

6.159.1 Constructor & Destructor Documentation

6.159.1.1 decaf::security::cert::CertificateException::CertificateException () throw () [inline]

Default Constructor.

6.159.1.2 decaf::security::cert::CertificateException::CertificateException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.159.1.3 decaf::security::cert::CertificateException::CertificateException (const CertificateException & ex) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.159.1.4 `decaf::security::cert::CertificateException::CertificateException (const char * file, const int lineNumber, const char * msg, ...) throw ()`
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.159.1.5 `virtual decaf::security::cert::CertificateException::~~CertificateException () throw ()` [inline, virtual]

6.159.2 Member Function Documentation

6.159.2.1 `virtual CertificateException* decaf::security::cert::CertificateException::clone () const`
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1708).

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 973), `decaf::security::cert::CertificateExpiredException` (p. 977), `decaf::security::cert::CertificateNotYetValidException` (p. 979), and `decaf::security::cert::CertificateParsingException` (p. 981).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

6.160 decaf::security::cert::CertificateExpiredException Class Reference

#include <src/main/decaf/security/cert/CertificateExpiredException.h>Inheritance diagram for decaf::security::cert::CertificateExpiredException:

Public Member Functions

- **CertificateExpiredException** () throw ()
Default Constructor.
- **CertificateExpiredException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()
Copy Constructor.
- **CertificateExpiredException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateExpiredException** * clone () const
Clones this exception.
- virtual ~**CertificateExpiredException** () throw ()

6.160.1 Constructor & Destructor Documentation

6.160.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException () throw () [inline]

Default Constructor.

6.160.1.2 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.160.1.3 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const CertificateExpiredException & ex) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.160.1.4 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.160.1.5 `virtual decaf::security::cert::CertificateExpiredException::~~CertificateExpiredException () throw ()` [inline, virtual]

6.160.2 Member Function Documentation

6.160.2.1 `virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone () const` [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.975).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateExpiredException.h`

6.161 decaf::security::cert::CertificateNotYetValidException Class Reference

#include <src/main/decaf/security/cert/CertificateNotYetValidException.h> Inheritance diagram for decaf::security::cert::CertificateNotYetValidException:

Public Member Functions

- **CertificateNotYetValidException** () throw ()
Default Constructor.
- **CertificateNotYetValidException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()
Copy Constructor.
- **CertificateNotYetValidException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateNotYetValidException** * clone () const
Clones this exception.
- virtual ~**CertificateNotYetValidException** () throw ()

6.161.1 Constructor & Destructor Documentation

6.161.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException () throw () [inline]

Default Constructor.

6.161.1.2 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.161.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const CertificateNotYetValidException & ex) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.161.1.4 `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.161.1.5 `virtual decaf::security::cert::CertificateNotYetValidException::~~CertificateNotYetValidException () throw () [inline, virtual]`

6.161.2 Member Function Documentation

6.161.2.1 `virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.975).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateNotYetValidException.h`

6.162 decaf::security::cert::CertificateParsingException Class Reference

#include <src/main/decaf/security/cert/CertificateParsingException.h> Inheritance diagram for decaf::security::cert::CertificateParsingException:

Public Member Functions

- **CertificateParsingException** () throw ()
Default Constructor.
- **CertificateParsingException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()
Copy Constructor.
- **CertificateParsingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateParsingException** * clone () const
Clones this exception.
- virtual ~**CertificateParsingException** () throw ()

6.162.1 Constructor & Destructor Documentation

6.162.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException () throw () [inline]

Default Constructor.

6.162.1.2 decaf::security::cert::CertificateParsingException::CertificateParsingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.162.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException (const CertificateParsingException & ex) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.162.1.4 `decaf::security::cert::CertificateParsingException::CertificateParsingException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.162.1.5 `virtual decaf::security::cert::CertificateParsingException::~~CertificateParsingException () throw () [inline, virtual]`

6.162.2 Member Function Documentation

6.162.2.1 `virtual CertificateParsingException* decaf::security::cert::CertificateParsingException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.975).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateParsingException.h`

6.163 decaf::lang::Character Class Reference

#include <src/main/decaf/lang/Character.h> Inheritance diagram for decaf::lang::Character:

Public Member Functions

- **Character** (char value)
- virtual int **compareTo** (const **Character** &c) const
*Compares this **Character** (p. 982) instance with another.*
- virtual bool **operator==** (const **Character** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Character** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const char &c) const
*Compares this **Character** (p. 982) instance with a char type.*
- virtual bool **operator==** (const char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Character** **valueOf** (char value)
*Returns a **Character** (p. 982) instance representing the specified char value.*
- static bool **isWhitespace** (char c)
Indicates whether or not the given character is considered whitespace.
- static bool **isDigit** (char c)
Indicates whether or not the given character is a digit.
- static bool **isLowerCase** (char c)
Indicates whether or not the given character is a lower case character.
- static bool **isUpperCase** (char c)
Indicates whether or not the given character is a upper case character.
- static bool **isLetter** (char c)
Indicates whether or not the given character is a letter.
- static bool **isLetterOrDigit** (char c)
Indicates whether or not the given character is either a letter or a digit.
- static bool **isISOControl** (char c)
Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.
- static int **digit** (char c, int radix)
Returns the numeric value of the character ch in the specified radix.

Static Public Attributes

- static const int **MIN_RADIX** = 2
The minimum radix available for conversion to and from strings.
- static const int **MAX_RADIX** = 36
The maximum radix available for conversion to and from strings.
- static const char **MIN_VALUE** = (char)0x7F
The minimum value that a signed char can take on.
- static const char **MAX_VALUE** = (char)0x80
The maximum value that a signed char can take on.
- static const int **SIZE** = 8
The size of the primitive charactor in bits.

6.163.1 Constructor & Destructor Documentation

6.163.1.1 `decaf::lang::Character::Character (char value)`

Parameters:

value - char to wrap.

6.163.2 Member Function Documentation

6.163.2.1 `virtual unsigned char decaf::lang::Character::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2432).

6.163.2.2 `virtual int decaf::lang::Character::compareTo (const char & c) const [inline, virtual]`

Compares this **Character** (p. 982) instance with a char type.

Parameters:

c - the char instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< char >` (p. 1083).

6.163.2.3 `virtual int decaf::lang::Character::compareTo (const Character & c) const [inline, virtual]`

Compares this **Character** (p. 982) instance with another.

Parameters:

c - the **Character** (p. 982) instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< Character >` (p. 1083).

6.163.2.4 static int decaf::lang::Character::digit (char *c*, int *radix*) [static]

Returns the numeric value of the character *ch* in the specified radix. If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

* The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. * The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than `radix + 'A' - 10`. In this case, `ch - 'A' + 10` is returned. * The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than `radix + 'a' - 10`. In this case, `ch - 'a' + 10` is returned.

Parameters:

c - the char to be converted
radix - the radix of the number

Returns:

the numeric value of the number represented in the given radix

6.163.2.5 virtual double decaf::lang::Character::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.163.2.6 bool decaf::lang::Character::equals (const char & *c*) const [inline, virtual]**Returns:**

true if the two **Character**s have the same value.

Implements **decaf::lang::Comparable< char >** (p. 1084).

6.163.2.7 bool decaf::lang::Character::equals (const Character & *c*) const [inline, virtual]**Returns:**

true if the two **Character** (p. 982) Objects have the same value.

Implements **decaf::lang::Comparable< Character >** (p. 1084).

6.163.2.8 virtual float decaf::lang::Character::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.163.2.9 `virtual int decaf::lang::Character::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.163.2.10 `static bool decaf::lang::Character::isDigit (char c) [inline, static]`

Indicates whether or not the given character is a digit.

6.163.2.11 `static bool decaf::lang::Character::isISOControl (char c) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

Parameters:

c - the character, including supplementary characters

Returns:

true if the char is an ISO control character

6.163.2.12 `static bool decaf::lang::Character::isLetter (char c) [inline, static]`

Indicates whether or not the given character is a letter.

6.163.2.13 `static bool decaf::lang::Character::isLetterOrDigit (char c) [inline, static]`

Indicates whether or not the given character is either a letter or a digit.

6.163.2.14 `static bool decaf::lang::Character::isLowerCase (char c) [inline, static]`

Indicates whether or not the given character is a lower case character.

6.163.2.15 `static bool decaf::lang::Character::isUpperCase (char c) [inline, static]`

Indicates whether or not the given character is a upper case character.

6.163.2.16 `static bool decaf::lang::Character::isWhitespace (char c) [inline, static]`

Indicates whether or not the given character is considered whitespace.

6.163.2.17 `virtual long long decaf::lang::Character::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.163.2.18 `virtual bool decaf::lang::Character::operator< (const char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1084).

6.163.2.19 `virtual bool decaf::lang::Character::operator< (const Character & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Character >** (p. 1084).

6.163.2.20 `virtual bool decaf::lang::Character::operator== (const char & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1085).

6.163.2.21 **virtual bool decaf::lang::Character::operator== (const Character & c)**
const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Character >** (p. 1085).

6.163.2.22 **virtual short decaf::lang::Character::shortValue () const** [inline, virtual]

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2434).

6.163.2.23 **std::string decaf::lang::Character::toString () const**

Returns:

this **Character** (p. 982) Object as a String Representation

6.163.2.24 **static Character decaf::lang::Character::valueOf (char value)** [inline, static]

Returns a **Character** (p. 982) instance representing the specified char value.

Parameters:

value - the primitive char to wrap.

Returns:

a new Character instance that wraps this value.

6.163.3 Field Documentation

6.163.3.1 `const int decaf::lang::Character::MAX_RADIX = 36` [static]

The maximum radix available for conversion to and from strings.

6.163.3.2 `const char decaf::lang::Character::MAX_VALUE = (char)0x80` [static]

The maximum value that a signed char can take on.

6.163.3.3 `const int decaf::lang::Character::MIN_RADIX = 2` [static]

The minimum radix available for conversion to and from strings.

6.163.3.4 `const char decaf::lang::Character::MIN_VALUE = (char)0x7F` [static]

The minimum value that a signed char can take on.

6.163.3.5 `const int decaf::lang::Character::SIZE = 8` [static]

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Character.h`

6.164 decaf::internal::nio::CharArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/CharArrayBuffer.h> Inheritance diagram for decaf::internal::nio::CharArrayBuffer:

Public Member Functions

- **CharArrayBuffer** (std::size_t capacity, bool **readOnly**=false)

*Creates a **CharArrayBuffer** (p. 990) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **CharArrayBuffer** (char *array, std::size_t **offset**, std::size_t capacity, bool **readOnly**=false) throw (decaf::lang::exceptions::NullPointerException)

*Creates a **CharArrayBuffer** (p. 990) object that wraps the given array.*
- **CharArrayBuffer** (ByteArrayPerspective &array, std::size_t **offset**, std::size_t length, bool **readOnly**=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset.*
- **CharArrayBuffer** (const **CharArrayBuffer** &other)

*Create a **CharArrayBuffer** (p. 990) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~**CharArrayBuffer** ()
- virtual char * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the character array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual CharBuffer * **asReadOnlyBuffer** () const

Creates a new, read-only char buffer that shares this buffer's content.
- virtual CharBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

Compacts this buffer.
- virtual CharBuffer * **duplicate** ()

Creates a new char buffer that shares this buffer's content.
- virtual char **get** () throw (decaf::nio::BufferUnderflowException)

Relative get method.
- virtual char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible char array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual CharBuffer & **put** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given char into this buffer at the current position, and then increments the position.
- virtual CharBuffer & **put** (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given char into this buffer at the given index.
- virtual CharBuffer * **slice** () const
Creates a new CharBuffer whose content is a shared subsequence of this buffer's content.
- virtual lang::CharSequence * **subSequence** (std::size_t start, std::size_t end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteArrayBuffer** (p. 870) as Read-Only.*

Protected Attributes

- bool **readOnly**
- internal::nio::ByteArrayPerspective * **_array**
- std::size_t **offset**

6.164.1 Constructor & Destructor Documentation

6.164.1.1 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (std::size_t *capacity*, bool *readOnly* = false)

Creates a **CharArrayBuffer** (p. 990) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

- capacity* - size of the array, this is the limit we read and write to.
- readOnly* - should this buffer be read-only, default as false

6.164.1.2 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (char * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)`

Creates a **CharArrayBuffer** (p. 990) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

- array* - array to wrap
- offset* - the position that is this buffers start pos.
- capacity* - size of the array, this is the limit we read and write to.
- readOnly* - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.164.1.3 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t length, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset. The capacity and limit of the new **CharArrayBuffer** (p. 990) will be that of the remaining capacity of the passed buffer.

Parameters:

- array* - the **ByteArrayPerspective** (p. 908) to wrap
- offset* - the offset into array where the buffer starts
- length* - the length of the array we are wrapping or limit.
- readOnly* - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.164.1.4 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & other)`

Create a **CharArrayBuffer** (p. 990) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.

Parameters:

- other* - the **CharArrayBuffer** (p. 990) this one is to mirror.

6.164.1.5 virtual decaf::internal::nio::CharArrayBuffer::~~CharArrayBuffer ()
[virtual]

6.164.2 Member Function Documentation

6.164.2.1 virtual char* decaf::internal::nio::CharArrayBuffer::array ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]

Returns the character array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements decaf::nio::CharBuffer (p. 1003).

6.164.2.2 virtual std::size_t decaf::internal::nio::CharArrayBuffer::arrayOffset
() throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements decaf::nio::CharBuffer (p. 1003).

6.164.2.3 virtual CharBuffer* de-
caf::internal::nio::CharArrayBuffer::asReadOnlyBuffer ()
const [virtual]

Creates a new, read-only char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 1003).

**6.164.2.4 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]**

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this CharBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 1004).

**6.164.2.5 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::duplicate ()
[virtual]**

Creates a new char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new char Buffer which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1005).

**6.164.2.6 virtual char decaf::internal::nio::CharArrayBuffer::get (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]**

Absolute get method. Reads the char at the given index.

Parameters:

index - the index in the Buffer where the char is to be read

Returns:

the char that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::CharBuffer** (p. 1006).

6.164.2.7 virtual char decaf::internal::nio::CharArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the character at this buffer's current position, and then increments the position.

Returns:

the char at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::CharBuffer** (p. 1006).

6.164.2.8 virtual bool decaf::internal::nio::CharArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible char array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 1007).

6.164.2.9 virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 806).

6.164.2.10 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given char into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data
value - the char to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 1009).

6.164.2.11 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given char into this buffer at the current position, and then increments the position.

Parameters:

value - the char value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 1009).

6.164.2.12 `virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.164.2.13 `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice () const [virtual]`

Creates a new CharBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create CharBuffer which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1011).

6.164.2.14 `virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (std::size_t start, std::size_t end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position. The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 807) + start, and its limit will be **position()** (p. 807) + end. The new Buffer will be read-only if, and only if, this buffer is read-only.

Parameters:

- start** - The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 808)
- end** - The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 808)

Returns:

The new character buffer, caller owns

Exceptions:

IndexOutOfBoundsException - If the preconditions on start and end fail

Implements **decaf::nio::CharBuffer** (p. 1012).

6.164.3 Field Documentation

6.164.3.1 `internal::nio::ByteArrayPerspective* decaf::internal::nio::CharArrayBuffer::_array [protected]`

6.164.3.2 `std::size_t decaf::internal::nio::CharArrayBuffer::offset [protected]`

6.164.3.3 `bool decaf::internal::nio::CharArrayBuffer::readOnly [protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/CharArrayBuffer.h`

6.165 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:.

```
#include <src/main/decaf/nio/CharBuffer.h>
Inheritance diagram for decaf::nio::CharBuffer:
```

Public Member Functions

- virtual **~CharBuffer** ()
- virtual std::string **toString** () const
- **CharBuffer** & **append** (char value) throw (BufferOverflowException, ReadOnlyBufferException)
Appends the specified character to this buffer.
- **CharBuffer** & **append** (const lang::CharSequence *value) throw (BufferOverflowException, ReadOnlyBufferException)
Appends the specified character sequence to this buffer.
- **CharBuffer** & **append** (const lang::CharSequence *value, std::size_t start, std::size_t end) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException)
Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.
- virtual char * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the character array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **CharBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only char buffer that shares this buffer's content.
- char **charAt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Reads the character at the given index relative to the current position.
- virtual **CharBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **CharBuffer** * **duplicate** ()=0
Creates a new char buffer that shares this buffer's content.
- virtual char **get** ()=0 throw (BufferUnderflowException)
Relative get method.

- virtual char **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **CharBuffer** & **get** (std::vector< char > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **CharBuffer** & **get** (char *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible char array.
- std::size_t **length** () const
Returns the length of this character buffer.
- **CharBuffer** & **put** (**CharBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the chars remaining in the given source buffer into this buffer.
- **CharBuffer** & **put** (const char *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers chars into this buffer from the given source array.
- **CharBuffer** & **put** (std::vector< char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source char array into this buffer.
- virtual **CharBuffer** & **put** (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given char into this buffer at the current position, and then increments the position.
- virtual **CharBuffer** & **put** (std::size_t index, char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given char into this buffer at the given index.
- **CharBuffer** & **put** (const std::string &src, std::size_t start, std::size_t end) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)
Relative bulk put method (optional operation).
- **CharBuffer** & **put** (const std::string &src) throw (BufferOverflowException, ReadOnlyBufferException)
Relative bulk put method (optional operation).
- virtual std::size_t **read** (**CharBuffer** *target) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException)
Attempts to read characters into the specified character buffer.

- virtual **lang::CharSequence * subSequence** (std::size_t start, std::size_t end) const =0
throw (decaf::lang::exceptions::IndexOutOfBounds Exception)
Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.
- virtual **CharBuffer * slice** () const =0
*Creates a new **CharBuffer** (p. 998) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **CharBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **CharBuffer** &value) const
- virtual bool **operator==** (const **CharBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **CharBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **CharBuffer * allocate** (std::size_t capacity)
Allocates a new character buffer.
- static **CharBuffer * wrap** (char *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointer Exception)
*Wraps the passed buffer with a new **CharBuffer** (p. 998).*
- static **CharBuffer * wrap** (std::vector< char > &buffer)
*Wraps the passed STL char Vector in a **CharBuffer** (p. 998).*

Protected Member Functions

- **CharBuffer** (std::size_t capacity)
*Creates a **CharBuffer** (p. 998) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.165.1 Detailed Description

This class defines four categories of operations upon character buffers: o Absolute and relative get and put methods that read and write single characters; o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer. o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the CharSequence interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package decaf.util.regex.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text/").put(subtype).put("; charset=").put(enc);
```

6.165.2 Constructor & Destructor Documentation

6.165.2.1 decaf::nio::CharBuffer::CharBuffer (std::size_t *capacity*) [protected]

Creates a **CharBuffer** (p. 998) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.165.2.2 virtual decaf::nio::CharBuffer::~~CharBuffer () [inline, virtual]

6.165.3 Member Function Documentation

6.165.3.1 static CharBuffer* decaf::nio::CharBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new character buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Char buffer in chars (1 byte).

Returns:

the **CharBuffer** (p. 998) that was allocated, caller owns.

6.165.3.2 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*, std::size_t *start*, std::size_t *end*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException) [virtual]

Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

Parameters:

value - the CharSequence to append.
start - the index to start appending from.
end - the index to append to.

Returns:

a reference to this modified **CharBuffer** (p. 998)

Exceptions:

BufferOverflowException (p. 834) if there is no more space
ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.
IndexOutOfBoundsException if start > end, or > length of sequence.

Implements **decaf::lang::Appendable** (p. 626).

6.165.3.3 **CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence *
value) throw (BufferOverflowException, ReadOnlyBufferException)**
[virtual]

Appends the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

Parameters:

value - the CharSequence to append.

Returns:

a reference to this modified **CharBuffer** (p. 998)

Exceptions:

BufferOverflowException (p. 834) if there is no more space
ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

Implements **decaf::lang::Appendable** (p. 627).

6.165.3.4 **CharBuffer& decaf::nio::CharBuffer::append (char value) throw (**
BufferOverflowException, ReadOnlyBufferException) [virtual]

Appends the specified character to this buffer.

Parameters:

value - the char to append.

Returns:

a reference to this modified **CharBuffer** (p. 998)

Exceptions:

BufferOverflowException (p. 834) if there is no more space

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

Implements **decaf::lang::Appendable** (p. 627).

6.165.3.5 `virtual char* decaf::nio::CharBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the character array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 803)

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 993).

6.165.3.6 `virtual std::size_t decaf::nio::CharBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 993).

6.165.3.7 `virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new

buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 993).

6.165.3.8 `char decaf::nio::CharBuffer::charAt (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [virtual]

Reads the character at the given index relative to the current position.

Parameters:

index - The index of the character to be read relative to position

Returns:

The character at index **position()** (p. 807) + index

Exceptions:

IndexOutOfBoundsException

Implements **decaf::lang::CharSequence** (p. 1014).

6.165.3.9 `virtual CharBuffer& decaf::nio::CharBuffer::compact () throw (ReadOnlyBufferException)` [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \mathbf{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index **limit()** (p. 807) - 1 is copied to index $n = \mathbf{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **CharBuffer** (p. 998)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 994).

6.165.3.10 virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.165.3.11 virtual CharBuffer* decaf::nio::CharBuffer::duplicate () [pure virtual]

Creates a new char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new char **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 994).

6.165.3.12 virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.165.3.13 CharBuffer& decaf::nio::CharBuffer::get (char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies `length` chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than *length* chars remaining in this buffer

NullPointerException if the passed buffer is null.

6.165.3.14 CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns:

a reference to this **CharBuffer** (p. 998)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than *length* chars remaining in this buffer

6.165.3.15 virtual char decaf::nio::CharBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the char at the given index.

Parameters:

index - the index in the **Buffer** (p. 803) where the char is to be read

Returns:

the char that is located at the given index

Exceptions:

IndexOutOfBoundsException - If *index* is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 994).

6.165.3.16 virtual char decaf::nio::CharBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the character at this buffer's current position, and then increments the position.

Returns:

the char at the current position

Exceptions:

BufferUnderflowException (p. 837) if there no more data to return

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 995).

6.165.3.17 virtual bool decaf::nio::CharBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible char array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 995).

6.165.3.18 std::size_t decaf::nio::CharBuffer::length () const [inline, virtual]

Returns the length of this character buffer.

Returns:

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 1015).

6.165.3.19 virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & value) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.165.3.20 virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & value) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.165.3.21 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*) throw (BufferOverflowException, ReadOnlyBufferException)

Relative bulk put method (optional operation). This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form `dst.put(s)` behaves in exactly the same way as the invocation

Parameters:

src - the string to copy from

Returns:

a reference to this **CharBuffer** (p. 998)

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.165.3.22 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*, std::size_t *start*, std::size_t *end*) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)

Relative bulk put method (optional operation). This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if `end - start > remaining()` (p. 808), then no characters are transferred and a **BufferOverflowException** (p. 834) is thrown.

Returns:

a reference to this buffer

Otherwise, this method copies `n = end - start` characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by `n`.

Parameters:

src - the string to copy from

start - position in *src* to start from

end - the position in *src* to stop at

Returns:

a reference to this **CharBuffer** (p. 998)

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.165.3.23 virtual CharBuffer& decaf::nio::CharBuffer::put (std::size_t *index*, char *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given char into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the char to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 995).

6.165.3.24 virtual CharBuffer& decaf::nio::CharBuffer::put (char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters:

value - the char value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 996).

6.165.3.25 CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source char array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size())

Parameters:

buffer - The buffer whose contents are copied to this **CharBuffer** (p. 998)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.165.3.26 **CharBuffer& decaf::nio::CharBuffer::put (const char * *buffer*,
std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException,
ReadOnlyBufferException, lang::exceptions::NullPointerException)**

This method transfers chars into this buffer from the given source array. If there are more chars to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 808), then no chars are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters:

buffer- The array from which chars are to be read

offset- The offset within the array of the first char to be read;

length - The number of chars to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.165.3.27 **CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & *src*)
throw (BufferOverflowException, ReadOnlyBufferException,
lang::exceptions::IllegalArgumentException)**

This method transfers the chars remaining in the given source buffer into this buffer. If there are more chars remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 808), then no chars are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `n = src.remaining()` chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters:

src - the buffer to take chars from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining chars in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.165.3.28 `virtual std::size_t decaf::nio::CharBuffer::read (CharBuffer *
target) throw (decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
ReadOnlyBufferException) [virtual]`

Attempts to read characters into the specified character buffer. The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters:

target - the buffer to read characters into

Returns:

The number of characters added to the buffer, or string::npos if this source of characters is at its end

Exceptions:

NullPointerException - If target is Null

IllegalArgumentException - If target is this

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.165.3.29 `virtual CharBuffer* decaf::nio::CharBuffer::slice () const [pure
virtual]`

Creates a new **CharBuffer** (p. 998) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **CharBuffer** (p. 998) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 996).

6.165.3.30 `virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence (std::size_t start, std::size_t end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position. The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 807) + start, and its limit will be **position()** (p. 807) + end. The new **Buffer** (p. 803) will be read-only if, and only if, this buffer is read-only.

Parameters:

- start* - The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 808)
- end* - The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 808)

Returns:

The new character buffer, caller owns

Exceptions:

IndexOutOfBoundsException - If the preconditions on start and end fail

Implements **decaf::lang::CharSequence** (p. 1015).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 997).

6.165.3.31 `virtual std::string decaf::nio::CharBuffer::toString () const [virtual]`

Returns:

a std::string describing this object

Implements **decaf::lang::CharSequence** (p. 1015).

6.165.3.32 `static CharBuffer* decaf::nio::CharBuffer::wrap (std::vector< char > & buffer) [static]`

Wraps the passed STL char Vector in a **CharBuffer** (p. 998). The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be **buffer.size()**, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

- buffer* - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling **vector.resize(N)**.

Returns:

a new **CharBuffer** (p. 998) that is backed by buffer, caller owns.

6.165.3.33 `static CharBuffer* decaf::nio::CharBuffer::wrap (char *
 array, std::size_t offset, std::size_t length) throw (
 lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **CharBuffer** (p. 998). The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

- array* - The array that will back the new buffer
- offset* - The offset of the subarray to be used
- length* - The length of the subarray to be used

Returns:

- a new **CharBuffer** (p. 998) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/CharBuffer.h`

6.166 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 1014) is a readable sequence of char values.

#include <src/main/decaf/lang/CharSequence.h> Inheritance diagram for decaf::lang::CharSequence:

Public Member Functions

- virtual **~CharSequence** ()
- virtual std::size_t **length** () const =0
- virtual char **charAt** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Returns the Char at the specified index so long as the index is not greater than the length of the sequence.
- virtual **CharSequence * subSequence** (std::size_t start, std::size_t end) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
*Returns a new **CharSequence** (p. 1014) that is a subsequence of this sequence.*
- virtual std::string **toString** () const =0

6.166.1 Detailed Description

A **CharSequence** (p. 1014) is a readable sequence of char values. This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 1014) should implement comparable, it is therefore up to the dervied classes that implement this interface to define equality, which implies that comparison of two **CharSequences** does not have a contract on equality.

6.166.2 Constructor & Destructor Documentation

6.166.2.1 virtual decaf::lang::CharSequence::~~CharSequence () [inline, virtual]

6.166.3 Member Function Documentation

6.166.3.1 virtual char decaf::lang::CharSequence::charAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters:

index - position to return the char at.

Returns:

the char at the given position

Exceptions:

IndexOutOfBoundsException if index is > than **length()** (p. 1015)

Implemented in **decaf::nio::CharBuffer** (p. 1004).

6.166.3.2 `virtual std::size_t decaf::lang::CharSequence::length () const [pure virtual]`

Returns:

the length of the underlying character sequence.

Implemented in **decaf::nio::CharBuffer** (p. 1007).

6.166.3.3 `virtual CharSequence* decaf::lang::CharSequence::subSequence (std::size_t start, std::size_t end) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Returns a new **CharSequence** (p. 1014) that is a subsequence of this sequence. The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

Parameters:

start - the start index, inclusive

end - the end index, exclusive

Returns:

a new **CharSequence** (p. 1014)

Exceptions:

IndexOutOfBoundsException if start or end > **length()** (p. 1015)

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 997), and **decaf::nio::CharBuffer** (p. 1012).

6.166.3.4 `virtual std::string decaf::lang::CharSequence::toString () const [pure virtual]`

Returns:

the string representation of this **CharSequence** (p. 1014)

Implemented in **decaf::nio::CharBuffer** (p. 1012).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

6.167 decaf::lang::exceptions::ClassCastException Class Reference

#include <src/main/decaf/lang/exceptions/ClassCastException.h> Inheritance diagram for decaf::lang::exceptions::ClassCastException:

Public Member Functions

- **ClassCastException** () throw ()
Default Constructor.
- **ClassCastException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()
Copy Constructor.
- **ClassCastException** (const std::exception *cause) throw ()
Constructor.
- **ClassCastException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ClassCastException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ClassCastException** * clone () const
Clones this exception.
- virtual ~**ClassCastException** () throw ()

6.167.1 Constructor & Destructor Documentation

6.167.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException () throw () [inline]

Default Constructor.

6.167.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.167.1.3 decaf::lang::exceptions::ClassCastException::ClassCastException (const ClassCastException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.167.1.4 decaf::lang::exceptions::ClassCastException::ClassCastException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.167.1.5 decaf::lang::exceptions::ClassCastException::ClassCastException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.167.1.6 decaf::lang::exceptions::ClassCastException::ClassCastException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.167.1.7 `virtual decaf::lang::exceptions::ClassCastException::~~ClassCastException
() throw () [inline, virtual]`

6.167.2 Member Function Documentation

6.167.2.1 `virtual ClassCastException* de-
caf::lang::exceptions::ClassCastException::clone () const
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/ClassCastException.h`

6.168 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

#include <src/main/decaf/io/Closeable.h> Inheritance diagram for decaf::io::Closeable:

Public Member Functions

- virtual `~Closeable ()`
- virtual void `close ()=0 throw (io::IOException)`
Closes this object and deallocates the appropriate resources.

6.168.1 Detailed Description

Interface for a class that implements the close method.

6.168.2 Constructor & Destructor Documentation

6.168.2.1 virtual `decaf::io::Closeable::~~Closeable ()` [inline, virtual]

6.168.3 Member Function Documentation

6.168.3.1 virtual void `decaf::io::Closeable::close () throw (io::IOException)` [pure virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

IOException (p. 1820) if an error occurs while closing.

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2788), `activemq::transport::failover::FailoverTransport` (p. 1615), `activemq::transport::inactivity::InactivityMonitor` (p. 1734), `activemq::transport::IOTransport` (p. 1825), `activemq::transport::mock::MockTransport` (p. 2373), `activemq::transport::tcp::TcpTransport` (p. 3161), `activemq::transport::TransportFilter` (p. 3283), `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2463), `decaf::internal::io::StandardErrorOutputStream` (p. 2995), `decaf::internal::io::StandardInputStream` (p. 3002), `decaf::internal::io::StandardOutputStream` (p. 3009), `decaf::io::BlockingByteArrayInputStream` (p. 726), `decaf::io::BufferedInputStream` (p. 811), `decaf::io::BufferedOutputStream` (p. 815), `decaf::io::ByteArrayInputStream` (p. 894), `decaf::io::ByteArrayOutputStream` (p. 902), `decaf::io::FilterInputStream` (p. 1633), `decaf::io::FilterOutputStream` (p. 1642), `decaf::net::BufferedSocket` (p. 818), `decaf::net::SocketInputStream` (p. 2979), `decaf::net::SocketOutputStream` (p. 2985), `decaf::net::TcpSocket` (p. 3154), and `decaf::util::logging::StreamHandler` (p. 3078).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Closeable.h`

6.169 cms::Closeable Class Reference

Interface for a class that implements the close method.

#include <src/main/cms/Closeable.h> Inheritance diagram for cms::Closeable:

Public Member Functions

- virtual `~Closeable ()`
- virtual void `close ()=0` throw (CMSEException)
Closes this object and deallocates the appropriate resources.

6.169.1 Detailed Description

Interface for a class that implements the close method. A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since:

1.0

6.169.2 Constructor & Destructor Documentation

6.169.2.1 virtual cms::Closeable::~~Closeable () [inline, virtual]

6.169.3 Member Function Documentation

6.169.3.1 virtual void cms::Closeable::close () throw (CMSEException) [pure virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

CMSEException (p. 1031) - If an error occurs while the resource is being closed.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 954), `activemq::cmsutil::CachedProducer` (p. 958), `activemq::cmsutil::PooledSession` (p. 2507), `activemq::commands::ActiveMQTempDestination` (p. 500), `activemq::core::ActiveMQConnection` (p. 236), `activemq::core::ActiveMQConsumer` (p. 266), `activemq::core::ActiveMQProducer` (p. 408), `activemq::core::ActiveMQSession` (p. 447), `cms::Connection` (p. 1132), and `cms::Session` (p. 2842).

The documentation for this class was generated from the following file:

- `src/main/cms/Closeable.h`

6.170 activemq::transport::failover::CloseTransportsTask Class Reference

#include <src/main/activemq/transport/failover/CloseTransportsTask.h> Inheritance diagram for activemq::transport::failover::CloseTransportsTask:

Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)
*Add a new **Transport** (p. 3273) to close.*
- virtual bool **isPending** () const
This Task is pending if there are transports in the Queue that need to be closed.
- virtual bool **iterate** ()
Return true until all transports have been closed and removed from the queue.

6.170.1 Constructor & Destructor Documentation

6.170.1.1 **activemq::transport::failover::CloseTransportsTask::CloseTransportsTask** () [inline]

6.170.1.2 **virtual**
activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask () [inline, virtual]

6.170.2 Member Function Documentation

6.170.2.1 **void** **activemq::transport::failover::CloseTransportsTask::add** (const **Pointer**< **Transport** > & *transport*)

Add a new **Transport** (p. 3273) to close.

6.170.2.2 **virtual bool** **activemq::transport::failover::CloseTransportsTask::isPending** () const [virtual]

This Task is pending if there are transports in the Queue that need to be closed.

Returns:

true if there is a transport in the queue that needs closed.

Implements **activemq::threads::CompositeTask** (p. 1090).

6.170.2.3 `virtual bool activemq::transport::failover::CloseTransportsTask::iterate ()` [virtual]

Return true until all transports have been closed and removed from the queue.

Implements `activemq::threads::Task` (p. 3148).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/CloseTransportsTask.h`

6.171 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p.1041) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p.1184) to operate on.

#include <src/main/activemq/cmsutil/CmsAccessor.h> Inheritance diagram for activemq::cmsutil::CmsAccessor:

Public Member Functions

- **CmsAccessor** ()
- virtual **~CmsAccessor** ()
- virtual **ResourceLifecycleManager** * **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** * **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (cms::ConnectionFactory *connectionFactory)
Set the ConnectionFactory to use for obtaining CMS Connections.
- virtual const cms::ConnectionFactory * **getConnectionFactory** () const
Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.
- virtual cms::ConnectionFactory * **getConnectionFactory** ()
Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.
- virtual void **setSessionAcknowledgeMode** (cms::Session::AcknowledgeMode sessionAcknowledgeMode)
Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.
- virtual cms::Session::AcknowledgeMode **getSessionAcknowledgeMode** () const
Return the acknowledgment mode for CMS sessions.

Protected Member Functions

- virtual void **init** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Initializes this object and prepares it for use.
- virtual void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Shuts down this object and destroys any allocated resources.
- virtual cms::Connection * **createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Create a CMS Connection via this template's ConnectionFactory.
- virtual cms::Session * **createSession** (cms::Connection *con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

Create a CMS Session for the given Connection.

- virtual void **checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException)

Verifies that the connection factory is valid.

6.171.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p.1041) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p.1184) to operate on. The subclass **activemq.cmsutil.CmsDestinationAccessor** (p.1028) adds further, destination-related properties.

Not intended to be used directly.

See also:

activemq.cmsutil.CmsDestinationAccessor (p.1028)
activemq.cmsutil.CmsTemplate (p.1041)

6.171.2 Constructor & Destructor Documentation

6.171.2.1 **activemq::cmsutil::CmsAccessor::CmsAccessor** ()

6.171.2.2 **virtual activemq::cmsutil::CmsAccessor::~~CmsAccessor** () [virtual]

6.171.3 Member Function Documentation

6.171.3.1 **virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Verifies that the connection factory is valid.

6.171.3.2 **virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

Returns:

the new CMS Connection

Exceptions:

cms::CMSException (p. 1031) if thrown by CMS API methods

6.171.3.3 `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Create a CMS Session for the given Connection.

Parameters:

con the CMS Connection to create a Session for

Returns:

the new CMS Session

Exceptions:

cms::CMSException (p. 1031) if thrown by CMS API methods

6.171.3.4 `virtual void activemq::cmsutil::CmsAccessor::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]`

Shuts down this object and destroys any allocated resources.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1029), and `activemq::cmsutil::CmsTemplate` (p.1044).

6.171.3.5 `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.171.3.6 `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () const [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.171.3.7 `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () const [inline, virtual]`

6.171.3.8 `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () [inline, virtual]`

6.171.3.9 `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode () const [inline, virtual]`

Return the acknowledgment mode for CMS sessions.

Returns:

the acknowledgment mode applied by this accessor

6.171.3.10 `virtual void activemq::cmsutil::CmsAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]`

Initializes this object and prepares it for use. This should be called before any other methods are called. This version does nothing.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.1029), and `activemq::cmsutil::CmsTemplate` (p.1047).

6.171.3.11 `virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * connectionFactory) [inline, virtual]`

Set the ConnectionFactory to use for obtaining CMS Connections.

6.171.3.12 `virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode (cms::Session::AcknowledgeMode sessionAcknowledgeMode) [inline, virtual]`

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message. Default is AUTO_ACKNOWLEDGE.

Parameters:

sessionAcknowledgeMode the acknowledgment mode

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsAccessor.h`

6.172 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the `CmsAccessor` (p. 1024) to add support for resolving destination names.

#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h> Inheritance diagram for `activemq::cmsutil::CmsDestinationAccessor`:

Public Member Functions

- `CmsDestinationAccessor ()`
- `virtual ~CmsDestinationAccessor ()`
- `virtual bool isPubSubDomain () const`
- `virtual void setPubSubDomain (bool pubSubDomain)`
- `virtual DestinationResolver * getDestinationResolver ()`
- `virtual const DestinationResolver * getDestinationResolver () const`
- `virtual void setDestinationResolver (DestinationResolver *destRes)`

Protected Member Functions

- `virtual void init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)`
Initializes the destination resolver.
- `virtual void destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)`
Calls `destroy()` (p. 1029) on the destination resolver.
- `virtual cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)`
Resolves the destination via the `DestinationResolver` (p. 1509).
- `virtual void checkDestinationResolver () throw (decaf::lang::exceptions::IllegalStateException)`
Verifies that the destination resolver is valid.

6.172.1 Detailed Description

Extends the `CmsAccessor` (p. 1024) to add support for resolving destination names. Not intended to be used directly.

See also:

`CmsTemplate` (p. 1041)
`CmsAccessor` (p. 1024)

6.172.2 Constructor & Destructor Documentation

6.172.2.1 `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ()`

6.172.2.2 `virtual
activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor ()
[virtual]`

6.172.3 Member Function Documentation

6.172.3.1 `virtual void ac-
tivemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver ()
throw (decaf::lang::exceptions::IllegalStateException) [protected,
virtual]`

Verifies that the destination resolver is valid.

6.172.3.2 `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy () throw
(cms::CMSException, decaf::lang::exceptions::IllegalStateException)
[protected, virtual]`

Calls `destroy()` (p. 1029) on the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 1026).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1044).

6.172.3.3 `virtual const DestinationResolver* ac-
tivemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()
const [inline, virtual]`

6.172.3.4 `virtual DestinationResolver* ac-
tivemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()
[inline, virtual]`

6.172.3.5 `virtual void activemq::cmsutil::CmsDestinationAccessor::init () throw
(cms::CMSException, decaf::lang::exceptions::IllegalStateException)
[protected, virtual]`

Initializes the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 1027).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1047).

6.172.3.6 virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain () const
[inline, virtual]

6.172.3.7 virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session * *session*, const std::string & *destName*) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
[protected, virtual]

Resolves the destination via the DestinationResolver (p. 1509).

Parameters:

session the session

destName the name of the destination.

Returns:

the destination

Exceptions:

cms::CMSException (p. 1031) if resolution failed.

decaf::lang::exceptions::IllegalStateException (p. 1726) if the destination resolver property is NULL.

6.172.3.8 virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver * *destRes*) [inline, virtual]

6.172.3.9 virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1052).

Referenced by **activemq::cmsutil::CmsTemplate::setPubSubDomain()**.

The documentation for this class was generated from the following file:

- **src/main/activemq/cmsutil/CmsDestinationAccessor.h**

6.173 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

#include <src/main/cms/CMSException.h> Inheritance diagram for cms::CMSException:

Public Member Functions

- **CMSException** () throw ()
- **CMSException** (const **CMSException** &ex) throw ()
- **CMSException** (const std::string &message, const std::exception *cause) throw ()
- **CMSException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSException** () throw ()
- virtual std::string **getMessage** () const
Gets the cause of the error.
- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual void **printStackTrace** () const
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- virtual const char * **what** () const throw ()
*Overloads the std::exception **what**() (p. 1033) function to return the cause of the exception.*

6.173.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes. This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type `std::exception` and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSEException** (p. 1031). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSEException** (p. 1031) instances.

Since:

1.0

6.173.2 Constructor & Destructor Documentation

6.173.2.1 `cms::CMSEException::CMSEException () throw ()`

6.173.2.2 `cms::CMSEException::CMSEException (const CMSEException & ex) throw ()`

6.173.2.3 `cms::CMSEException::CMSEException (const std::string & message, const std::exception * cause) throw ()`

6.173.2.4 `cms::CMSEException::CMSEException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`

6.173.2.5 `virtual cms::CMSEException::~~CMSEException () throw () [virtual]`

6.173.3 Member Function Documentation

6.173.3.1 `virtual const std::exception* cms::CMSEException::getCause () const [virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

6.173.3.2 `virtual std::string cms::CMSEException::getMessage () const [virtual]`

Gets the cause of the error.

Returns:

string errors message

6.173.3.3 `virtual std::vector< std::pair< std::string, int> > cms::CMSEException::getStackTrace () const [virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns:

vector containing stack trace strings

6.173.3.4 virtual std::string cms::CMSException::getStackTraceString () const [virtual]

Gets the stack trace as one contiguous string.

Returns:

string with formatted stack trace data

6.173.3.5 virtual void cms::CMSException::printStackTrace (std::ostream & *stream*) const [virtual]

Prints the stack trace to the given output stream.

Parameters:

stream the target output stream.

6.173.3.6 virtual void cms::CMSException::printStackTrace () const [virtual]

Prints the stack trace to std::err.

6.173.3.7 virtual void cms::CMSException::setMark (const char * *file*, const int *lineNumber*) [virtual]

Adds a file/line number to the stack trace.

Parameters:

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

6.173.3.8 virtual const char* cms::CMSException::what () const throw () [virtual]

Overloads the std::exception **what()** (p. 1033) function to return the cause of the exception.

Returns:

const char pointer to error message

The documentation for this class was generated from the following file:

- `src/main/cms/CMSException.h`

6.174 activemq::util::CMSExceptionSupport Class Reference

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

Public Member Functions

- virtual `~CMSExceptionSupport()`

Static Public Member Functions

- static `cms::CMSException create (const std::string &msg, const decaf::lang::Exception &cause)`
- static `cms::CMSException create (const decaf::lang::Exception &cause)`
- static `cms::MessageEOFException createMessageEOFException (const decaf::lang::Exception &cause)`
- static `cms::MessageFormatException createMessageFormatException (const decaf::lang::Exception &cause)`

6.174.1 Constructor & Destructor Documentation

6.174.1.1 virtual `activemq::util::CMSExceptionSupport::~~CMSExceptionSupport()` [virtual]

6.174.2 Member Function Documentation

6.174.2.1 static `cms::CMSException activemq::util::CMSExceptionSupport::create (const decaf::lang::Exception & cause)` [static]

6.174.2.2 static `cms::CMSException activemq::util::CMSExceptionSupport::create (const std::string & msg, const decaf::lang::Exception & cause)` [static]

6.174.2.3 static `cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException (const decaf::lang::Exception & cause)` [static]

6.174.2.4 static `cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException (const decaf::lang::Exception & cause)` [static]

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CMSExceptionSupport.h`

6.175 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

#include <src/main/cms/CMSProperties.h> Inheritance diagram for cms::CMSProperties:

Public Member Functions

- virtual **~CMSProperties** ()
- virtual bool **isEmpty** () const =0
Returns true if the properties object is empty.
- virtual const char * **getProperty** (const std::string &name) const =0
Looks up the value for the given property.
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const =0
Looks up the value for the given property.
- virtual void **setProperty** (const std::string &name, const std::string &value)=0
Sets the value for a given property.
- virtual bool **hasProperty** (const std::string &name) const =0
Check to see if the Property exists in the set.
- virtual void **remove** (const std::string &name)=0
Removes the property with the given name.
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const =0
Method that serializes the contents of the property map to an array.
- virtual void **copy** (const **CMSProperties** *source)=0
Copies the contents of the given properties object to this one.
- virtual **CMSProperties** * **clone** () const =0
Clones this object.
- virtual void **clear** ()=0
Clears all properties from the map.
- virtual std::string **toString** () const =0
Formats the contents of the Properties Object into a string that can be logged, etc.

6.175.1 Detailed Description

Interface for a Java-like properties object. This is essentially a map of key-value string pairs.

Since:

1.1

6.175.2 Constructor & Destructor Documentation

6.175.2.1 virtual cms::CMSProperties::~~CMSProperties () [inline, virtual]

6.175.3 Member Function Documentation

6.175.3.1 virtual void cms::CMSProperties::clear () [pure virtual]

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 415).

6.175.3.2 virtual CMSProperties* cms::CMSProperties::clone () const [pure virtual]

Clones this object.

Returns:

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 415).

6.175.3.3 virtual void cms::CMSProperties::copy (const CMSProperties * *source*) [pure virtual]

Copies the contents of the given properties object to this one.

Parameters:

source The source properties object.

6.175.3.4 virtual std::string cms::CMSProperties::getProperty (const std::string & *name*, const std::string & *defaultValue*) const [pure virtual]

Looks up the value for the given property.

Parameters:

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns:

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in **activemq::util::ActiveMQProperties** (p. 416).

6.175.3.5 `virtual const char* cms::CMSProperties::getProperty (const std::string & name) const` [pure virtual]

Looks up the value for the given property.

Parameters:

name The name of the property to be looked up.

Returns:

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in `activemq::util::ActiveMQProperties` (p. 416).

6.175.3.6 `virtual bool cms::CMSProperties::hasProperty (const std::string & name) const` [pure virtual]

Check to see if the Property exists in the set.

Parameters:

name the name of the property to check

Returns:

true if property exists, false otherwise.

Implemented in `activemq::util::ActiveMQProperties` (p. 416).

6.175.3.7 `virtual bool cms::CMSProperties::isEmpty () const` [pure virtual]

Returns true if the properties object is empty.

Returns:

true if empty

Implemented in `activemq::util::ActiveMQProperties` (p. 417).

6.175.3.8 `virtual void cms::CMSProperties::remove (const std::string & name)` [pure virtual]

Removes the property with the given name.

Parameters:

name the name of the property to be removed.s

Implemented in `activemq::util::ActiveMQProperties` (p. 417).

6.175.3.9 `virtual void cms::CMSProperties::setProperty (const std::string & name, const std::string & value)` [pure virtual]

Sets the value for a given property. If the property already exists, overwrites the value.

Parameters:

name The name of the value to be written.

value The value to be written.

Implemented in `activemq::util::ActiveMQProperties` (p. 417).

6.175.3.10 `virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray () const` [pure virtual]

Method that serializes the contents of the property map to an array.

Returns:

list of pairs where the first is the name and the second is the value.

Implemented in `activemq::util::ActiveMQProperties` (p. 417).

6.175.3.11 `virtual std::string cms::CMSProperties::toString () const` [pure virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns:

string value of this object.

Implemented in `activemq::util::ActiveMQProperties` (p. 417).

The documentation for this class was generated from the following file:

- `src/main/cms/CMSProperties.h`

6.176 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

#include <src/main/cms/CMSSecurityException.h> Inheritance diagram for cms::CMSSecurityException:

Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

6.176.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client. It may also be thrown for any case where a security restriction prevents a method from completing.

Since:

1.3

6.176.2 Constructor & Destructor Documentation

- 6.176.2.1** cms::CMSSecurityException::CMSSecurityException () throw ()
- 6.176.2.2** cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & ex) throw ()
- 6.176.2.3** cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause) throw ()
- 6.176.2.4** cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()
- 6.176.2.5** virtual cms::CMSSecurityException::~~CMSSecurityException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/CMSSecurityException.h

6.177 activemq::cmsutil::CmsTemplate Class Reference

CmsTemplate (p. 1041) simplifies performing synchronous CMS operations.

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate:

Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory *connectionFactory)
- virtual ~**CmsTemplate** ()
- virtual void **setDefaultDestination** (cms::Destination *defaultDestination)
Sets the destination object to be used by default for send/receive operations.
- virtual const cms::Destination * **getDefaultDestination** () const
Retrieves the default destination to be used for send/receive operations.
- virtual cms::Destination * **getDefaultDestination** ()
Retrieves the default destination to be used for send/receive operations.
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)
Sets the name of the default destination to be used from send/receive operations.
- virtual const std::string **getDefaultDestinationName** () const
Gets the name of the default destination to be used for send/receive operations.
- virtual void **setPubSubDomain** (bool pubSubDomain)
Indicates whether the default destination is a topic (true) or a queue (false).
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)
Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

- virtual bool **isExplicitQosEnabled** () const
If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)
Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").
- virtual void **setDeliveryMode** (int deliveryMode)
Set the delivery mode to use when sending a message.
- virtual int **getDeliveryMode** () const
Return the delivery mode to use when sending a message.
- virtual void **setPriority** (int priority)
Set the priority of a message when sending.
- virtual int **getPriority** () const
Return the priority of a message when sending.
- virtual void **setTimeToLive** (long long timeToLive)
Set the time-to-live of the message when sending.
- virtual long long **getTimeToLive** () const
Return the time-to-live of the message when sending.
- virtual void **execute** (SessionCallback *action) throw (cms::CMSEException)
Executes the given action within a CMS Session.
- virtual void **execute** (ProducerCallback *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (cms::Destination *dest, ProducerCallback *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (const std::string &destinationName, ProducerCallback *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **send** (MessageCreator *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the default destination.
- virtual void **send** (cms::Destination *dest, MessageCreator *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the specified destination.
- virtual void **send** (const std::string &destinationName, MessageCreator *messageCreator) throw (cms::CMSEException)

Convenience method for sending a message to the specified destination.

- virtual **cms::Message** * **receive** () throw (cms::CMSEException)
Performs a synchronous read from the default destination.
- virtual **cms::Message** * **receive** (cms::Destination *destination) throw (cms::CMSEException)
Performs a synchronous read from the specified destination.
- virtual **cms::Message** * **receive** (const std::string &destinationName) throw (cms::CMSEException)
Performs a synchronous read from the specified destination.
- virtual **cms::Message** * **receiveSelected** (const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (cms::Destination *destination, const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (const std::string &destinationName, const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Static Public Attributes

- static const long long **RECEIVE_TIMEOUT_NO_WAIT** = -1
Timeout value indicating that a receive operation should check if a message is immediately available without blocking.
- static const long long **RECEIVE_TIMEOUT_INDEFINITE_WAIT** = 0
Timeout value indicating a blocking receive without timeout.
- static const int **DEFAULT_PRIORITY** = 4
Default message priority.
- static const long long **DEFAULT_TIME_TO_LIVE** = 0
My default, messages should live forever.

Protected Member Functions

- void **init** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
Initializes this object and prepares it for use.

- void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

Clears all internal resources.

Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

6.177.1 Detailed Description

CmsTemplate (p. 1041) simplifies performing synchronous CMS operations. This class is intended to be for CMS what Spring's **JmsTemplate** is for JMS. Provided with a CMS **ConnectionFactory**, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 1041) the user must first set the destination (either by name or by setting the destination object directly) and then call **init** to initialize the object for use.

CmsTemplate (p. 1041) allows the user to get access to a CMS **Session** through a user-defined **SessionCallback** (p. 2852). Similarly, if the user wants direct access to a CMS **MessageProducer**, it can provide a **ProducerCallback** (p. 2603). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the **send** methods.

See also:

SessionCallback (p. 2852)
ProducerCallback (p. 2603)
MessageCreator (p. 2218)

6.177.2 Constructor & Destructor Documentation

6.177.2.1 **activemq::cmsutil::CmsTemplate::CmsTemplate ()**

6.177.2.2 **activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * *connectionFactory*)**

6.177.2.3 **virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate () [virtual]**

6.177.3 Member Function Documentation

6.177.3.1 **void activemq::cmsutil::CmsTemplate::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]**

Clears all internal resources.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1029).

6.177.3.2 virtual void activemq::cmsutil::CmsTemplate::execute (const std::string & *destinationName*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters:

destinationName the name of the destination to send messages to (to internally be resolved to an actual destination)

action the action to perform

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs.

6.177.3.3 virtual void activemq::cmsutil::CmsTemplate::execute (cms::Destination * *dest*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters:

dest the destination to send messages to

action the action to perform

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs.

6.177.3.4 virtual void activemq::cmsutil::CmsTemplate::execute (ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters:

action the action to perform

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs.

6.177.3.5 virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action within a CMS Session.

Parameters:

action the action to perform within a CMS Session

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs.

6.177.3.6 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination ()`
[inline, virtual]

Retrieves the default destination to be used for send/receive operations.

Returns:

the default destination. Non-const version of this method.

6.177.3.7 `virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () const`
[inline, virtual]

Retrieves the default destination to be used for send/receive operations.

Returns:

the default destination. Const version of this method.

6.177.3.8 `virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName () const`
[inline, virtual]

Gets the name of the default destination to be used for send/receive operations. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Returns:

the default name of the destination for send/receive operations.

6.177.3.9 `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode () const`
[inline, virtual]

Return the delivery mode to use when sending a message.

6.177.3.10 `virtual int activemq::cmsutil::CmsTemplate::getPriority () const`
[inline, virtual]

Return the priority of a message when sending.

6.177.3.11 `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout () const` [inline, virtual]

6.177.3.12 `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive () const` [inline, virtual]

Return the time-to-live of the message when sending.

6.177.3.13 `void activemq::cmsutil::CmsTemplate::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)`
[protected, virtual]

Initializes this object and prepares it for use. This should be called before any other methods are called.

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1029).

6.177.3.14 `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled ()`
`const` [inline, virtual]

If "true", then the values of `deliveryMode`, `priority`, and `timeToLive` will be used when sending a message. Otherwise, the default values, that may be set administratively, will be used.

Returns:

true if overriding default values of QOS parameters (`deliveryMode`, `priority`, and `timeToLive`)

See also:

`setDeliveryMode` (p. 1051)
`setPriority` (p. 1052)
`setTimeToLive` (p. 1052)

6.177.3.15 `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled ()`
`const` [inline, virtual]

6.177.3.16 `virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled ()` `const`
[inline, virtual]

6.177.3.17 `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal ()` `const`
[inline, virtual]

6.177.3.18 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive`
`(const std::string & destinationName) throw (cms::CMSException)`
[virtual]

Performs a synchronous read from the specified destination.

Parameters:

destinationName the name of the destination to receive on (will be resolved to destination internally).

Returns:

the message

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs

6.177.3.19 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * destination) throw (cms::CMSEException) [virtual]`

Performs a synchronous read from the specified destination.

Parameters:

destination the destination to receive on

Returns:

the message

Exceptions:

cms::CMSEException (p. 1031) thrown if an error occurs

6.177.3.20 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive () throw (cms::CMSEException) [virtual]`

Performs a synchronous read from the default destination.

Returns:

the message

Exceptions:

cms::CMSEException (p. 1031) thrown if an error occurs

6.177.3.21 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & destinationName, const std::string & selector) throw (cms::CMSEException) [virtual]`

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters:

destinationName the name of the destination to receive on (will be resolved to destination internally).

selector the selector expression.

Returns:

the message

Exceptions:

cms::CMSEException (p. 1031) thrown if an error occurs

6.177.3.22 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (cms::Destination * destination, const std::string & selector) throw (cms::CMSException)` [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters:

destination the destination to receive on.

selector the selector expression.

Returns:

the message

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs

6.177.3.23 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & selector) throw (cms::CMSException)` [virtual]

Performs a synchronous read consuming only messages identified by the given selector.

Parameters:

selector the selector expression.

Returns:

the message

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs

6.177.3.24 `virtual void activemq::cmsutil::CmsTemplate::send (const std::string & destinationName, MessageCreator * messageCreator) throw (cms::CMSException)` [virtual]

Convenience method for sending a message to the specified destination.

Parameters:

destinationName The name of the destination to send to.

messageCreator Responsible for creating the message to be sent

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs.

6.177.3.25 `virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination * dest, MessageCreator * messageCreator) throw (cms::CMSEException) [virtual]`

Convenience method for sending a message to the specified destination.

Parameters:

dest The destination to send to

messageCreator Responsible for creating the message to be sent

Exceptions:

cms::CMSEException (p. 1031) thrown if an error occurs.

6.177.3.26 `virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * messageCreator) throw (cms::CMSEException) [virtual]`

Convenience method for sending a message to the default destination.

Parameters:

messageCreator Responsible for creating the message to be sent

Exceptions:

cms::CMSEException (p. 1031) thrown if an error occurs.

6.177.3.27 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination (cms::Destination * defaultDestination) [inline, virtual]`

Sets the destination object to be used by default for send/receive operations. If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

Parameters:

defaultDestination the default destination

6.177.3.28 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName (const std::string & defaultDestinationName) [inline, virtual]`

Sets the name of the default destination to be used from send/receive operations. Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Parameters:

defaultDestinationName the name of the destination for send/receive to by default.

6.177.3.29 virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode (int *deliveryMode*) [inline, virtual]

Set the delivery mode to use when sending a message. Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters:

deliveryMode the delivery mode to use

See also:

`isExplicitQosEnabled` (p. 1047)

6.177.3.30 virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent (bool *deliveryPersistent*) [inline, virtual]

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false"). This will set the delivery mode accordingly, to either "PERSISTENT" or "NON_PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

See also:

`setDeliveryMode(int)` (p. 1051)

6.177.3.31 virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled (bool *explicitQosEnabled*) [inline, virtual]

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also:

`setDeliveryMode` (p. 1051)

`setPriority` (p. 1052)

`setTimeToLive` (p. 1052)

- 6.177.3.32** `virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled (bool messageIdEnabled)` [inline, virtual]
- 6.177.3.33** `virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled (bool messageTimestampEnabled)` [inline, virtual]
- 6.177.3.34** `virtual void activemq::cmsutil::CmsTemplate::setNoLocal (bool noLocal)` [inline, virtual]
- 6.177.3.35** `virtual void activemq::cmsutil::CmsTemplate::setPriority (int priority)` [inline, virtual]

Set the priority of a message when sending. Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also:

`isExplicitQosEnabled` (p. 1047)

- 6.177.3.36** `virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain (bool pubSubDomain)` [inline, virtual]

Indicates whether the default destination is a topic (true) or a queue (false). Calling this method will set the `defaultDestination` property to NULL.

Parameters:

pubSubDomain indicates whether to use pub-sub messaging (topics).

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1030).

References `activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()`.

- 6.177.3.37** `virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout (long long receiveTimeout)` [inline, virtual]
- 6.177.3.38** `virtual void activemq::cmsutil::CmsTemplate::setTimeToLive (long long timeToLive)` [inline, virtual]

Set the time-to-live of the message when sending. Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters:

timeToLive the message's lifetime (in milliseconds)

See also:

`isExplicitQosEnabled` (p. 1047)

6.177.4 Friends And Related Function Documentation

6.177.4.1 friend class `ProducerExecutor` [friend]

6.177.4.2 friend class `ReceiveExecutor` [friend]

6.177.4.3 friend class `ResolveProducerExecutor` [friend]

6.177.4.4 friend class `ResolveReceiveExecutor` [friend]

6.177.4.5 friend class `SendExecutor` [friend]

6.177.5 Field Documentation

6.177.5.1 `const int activemq::cmsutil::CmsTemplate::DEFAULT_PRIORITY = 4`
[static]

Default message priority.

6.177.5.2 `const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE = 0` [static]

My default, messages should live forever.

6.177.5.3 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_INDEFINITE_WAIT = 0` [static]

Timeout value indicating a blocking receive without timeout.

6.177.5.4 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_NO_WAIT = -1` [static]

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.178 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h> Inheritance diagram for
decaf::util::Collection< E >:
```

Public Member Functions

- virtual **~Collection** ()
- virtual bool **add** (const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Returns true if this collection changed as a result of the call.
- virtual bool **addAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Adds all of the elements in the specified collection to this collection.
- virtual void **clear** ()=0 throw (lang::exceptions::UnsupportedOperationException)
Removes all of the elements from this collection (optional operation).
- virtual bool **contains** (const E &value) const =0 throw (lang::Exception)
Returns true if this collection contains the specified element.
- virtual bool **containsAll** (const **Collection**< E > &collection) const =0 throw (lang::Exception)
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (const **Collection**< E > &value) const =0
Compares the passed collection to this one, if they contain the same elements, i.e.
- virtual bool **isEmpty** () const =0
- virtual bool **remove** (const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from the collection.
- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes all this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Retains only the elements in this collection that are contained in the specified collection (optional operation).

- virtual std::size_t **size** () const =0

Returns the number of elements in this collection.

- virtual std::vector< E > **toArray** () const =0

Returns an array containing all of the elements in this collection.

6.178.1 Detailed Description

template<typename E> class decaf::util::Collection< E >

The root interface in the collection hierarchy. A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 1054) implementation classes (which typically implement **Collection** (p. 1054) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 1054), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 1054) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw UnsupportedOperationException if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an UnsupportedOperationException if the invocation would have no effect on the collection. For example, invoking the addAll(Collection) method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the equals method. For example, the specification for the contains(Object o) method says: "returns true if and only if this collection contains at least one element e such that (o==null ? e==null : o.equals(e))."

Since:

1.0

6.178.2 Constructor & Destructor Documentation

6.178.2.1 `template<typename E> virtual decaf::util::Collection< E >::~~Collection
() [inline, virtual]`

6.178.3 Member Function Documentation

6.178.3.1 `template<typename E> virtual bool decaf::util::Collection<
E >::add (const E & value) throw (
lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException,
lang::exceptions::IllegalStateException) [pure
virtual]`

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1054) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in `decaf::util::AbstractQueue< E >` (p. 163), `decaf::util::PriorityQueue< E >` (p. 2574), `decaf::util::StlList< E >` (p. 3022), `decaf::util::StlSet< E >` (p. 3053), `decaf::util::StlList< CompositeTask * >` (p. 3022), `decaf::util::StlList< URI >` (p. 3022), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3022), `decaf::util::StlList< PrimitiveValueNode >` (p. 3022), `decaf::util::StlList< Pointer< Command > >` (p. 3022), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3022), `decaf::util::StlSet< transport::TransportListener * >` (p. 3053), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3053), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3053).

```

6.178.3.2  template<typename E> virtual bool decaf::util::Collection<
           E >::addAll (const Collection< E > & collection) throw
           ( lang::exceptions::UnsupportedOperationException,
           lang::exceptions::IllegalArgumentException,
           lang::exceptions::IllegalStateException ) [pure
           virtual]

```

Adds all of the elements in the specified collection to this collection. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters:

collection - **Collection** (p. 1054) whose elements are added to this one.

Returns:

true if this collection changed as a result of the call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractCollection< E >** (p. 151), **decaf::util::AbstractQueue< E >** (p. 163), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 151), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 151), **decaf::util::AbstractCollection< CompositeTask * >** (p. 151), **decaf::util::AbstractCollection< URI >** (p. 151), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 151), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 151), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 151), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 151), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 151).

```

6.178.3.3  template<typename E> virtual void decaf::util::Collection< E >::clear
           () throw ( lang::exceptions::UnsupportedOperationException ) [pure
           virtual]

```

Removes all of the elements from this collection (optional operation). This collection will be empty after this method returns unless it throws an exception.

Exceptions:

UnsupportedOperationException

Implemented in **decaf::util::AbstractCollection< E >** (p. 151), **decaf::util::AbstractQueue< E >** (p. 164), **decaf::util::PriorityQueue< E >** (p. 2574), **decaf::util::StlList< E >** (p. 3024), **decaf::util::StlSet< E >** (p. 3054), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 151), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 151), **decaf::util::AbstractCollection< CompositeTask * >** (p. 151),

decaf::util::AbstractCollection< URI > (p.151), decaf::util::AbstractCollection< ActiveMQSession * > (p.151), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p.151), decaf::util::AbstractCollection< PrimitiveValueNode > (p.151), decaf::util::AbstractCollection< Pointer< Command > > (p.151), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p.151), decaf::util::StlList< CompositeTask * > (p.3024), decaf::util::StlList< URI > (p.3024), decaf::util::StlList< Pointer< DestinationInfo > > (p.3024), decaf::util::StlList< PrimitiveValueNode > (p.3024), decaf::util::StlList< Pointer< Command > > (p.3024), decaf::util::StlList< Pointer< BackupTransport > > (p.3024), decaf::util::StlSet< transport::TransportListener * > (p.3054), decaf::util::StlSet< Pointer< Synchronization > > (p.3054), and decaf::util::StlSet< ActiveMQSession * > (p.3054).

6.178.3.4 `template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & value) const throw (lang::Exception)` [pure virtual]

Returns true if this collection contains the specified element. More formally, returns true if and only if this collection contains at least one element *e* such that (*o*==null ? *e*==null : *o.equals(e)*).

Parameters:

value - value to check for presence in the collection

Returns:

true if there is at least one of the elements in the collection

Exceptions:

Exception

Implemented in decaf::util::AbstractCollection< E > (p.152), decaf::util::StlList< E > (p.3024), decaf::util::StlSet< E > (p.3055), decaf::util::AbstractCollection< transport::TransportListener * > (p.152), decaf::util::AbstractCollection< Pointer< Synchronization > > (p.152), decaf::util::AbstractCollection< CompositeTask * > (p.152), decaf::util::AbstractCollection< URI > (p.152), decaf::util::AbstractCollection< ActiveMQSession * > (p.152), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p.152), decaf::util::AbstractCollection< PrimitiveValueNode > (p.152), decaf::util::AbstractCollection< Pointer< Command > > (p.152), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p.152), decaf::util::StlList< CompositeTask * > (p.3024), decaf::util::StlList< URI > (p.3024), decaf::util::StlList< Pointer< DestinationInfo > > (p.3024), decaf::util::StlList< PrimitiveValueNode > (p.3024), decaf::util::StlList< Pointer< Command > > (p.3024), decaf::util::StlList< Pointer< BackupTransport > > (p.3024), decaf::util::StlSet< transport::TransportListener * > (p.3055), decaf::util::StlSet< Pointer< Synchronization > > (p.3055), and decaf::util::StlSet< ActiveMQSession * > (p.3055).

6.178.3.5 `template<typename E> virtual bool decaf::util::Collection< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception)` [pure virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters:

collection - Collection (p.1054) to compare to this one.

Exceptions:*Exception*

Implemented in `decaf::util::AbstractCollection< E >` (p. 152),
`decaf::util::AbstractCollection< transport::TransportListener * >`
 (p. 152), `decaf::util::AbstractCollection< Pointer< Synchronization > >`
 (p. 152), `decaf::util::AbstractCollection< CompositeTask * >` (p. 152),
`decaf::util::AbstractCollection< URI >` (p. 152), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 152),
`decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 152), `decaf::util::AbstractCollection< PrimitiveValueNode >`
 (p. 152), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 152), and
`decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 152).

6.178.3.6 `template<typename E> virtual bool decaf::util::Collection< E >::equals (const Collection< E > & value) const` [pure virtual]

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

Returns:

true if the Collections contain the same elements.

Implemented in `decaf::util::AbstractCollection< E >` (p. 153),
`decaf::util::AbstractCollection< transport::TransportListener * >`
 (p. 153), `decaf::util::AbstractCollection< Pointer< Synchronization > >`
 (p. 153), `decaf::util::AbstractCollection< CompositeTask * >` (p. 153),
`decaf::util::AbstractCollection< URI >` (p. 153), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 153),
`decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 153), `decaf::util::AbstractCollection< PrimitiveValueNode >`
 (p. 153), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 153), and
`decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 153).

6.178.3.7 `template<typename E> virtual bool decaf::util::Collection< E >::isEmpty () const` [pure virtual]

Returns:

true if this collection contains no elements.

Implemented in `decaf::util::AbstractCollection< E >` (p. 153), `decaf::util::StlList< E >` (p. 3025), `decaf::util::StlSet< E >` (p. 3055), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 153), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 153), `decaf::util::AbstractCollection< CompositeTask * >` (p. 153), `decaf::util::AbstractCollection< URI >` (p. 153), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 153), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 153), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 153), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 153), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 153), `decaf::util::StlList< CompositeTask * >` (p. 3025), `decaf::util::StlList< URI >` (p. 3025), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3025), `decaf::util::StlList< PrimitiveValueNode >` (p. 3025), `decaf::util::StlList< Pointer< Command > >` (p. 3025), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3025), `decaf::util::StlSet<`

`transport::TransportListener * >` (p. 3055), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3055), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3055).

6.178.3.8 `template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes a single instance of the specified element from the collection. More formally, removes an element *e* such that (*o*==null ? *e*==null : *o*.equals(*e*)), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters:

value - reference to the element to remove.

Returns:

true if the collection was changed

Exceptions:

UnsupportedOperationException

IllegalArgumentException

Implemented in `decaf::util::AbstractCollection< E >` (p. 155), `decaf::util::PriorityQueue< E >` (p. 2577), `decaf::util::StlList< E >` (p. 3027), `decaf::util::StlSet< E >` (p. 3056), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 155), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 155), `decaf::util::AbstractCollection< CompositeTask * >` (p. 155), `decaf::util::AbstractCollection< URI >` (p. 155), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 155), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 155), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 155), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 155), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 155), `decaf::util::StlList< CompositeTask * >` (p. 3027), `decaf::util::StlList< URI >` (p. 3027), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3027), `decaf::util::StlList< PrimitiveValueNode >` (p. 3027), `decaf::util::StlList< Pointer< Command > >` (p. 3027), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3027), `decaf::util::StlSet< transport::TransportListener * >` (p. 3056), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3056), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3056).

6.178.3.9 `template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes all this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the specified collection.

Parameters:

collection - The **Collection** (p. 1054) whose elements are to be removed

Returns:

true if the collection changed as a result of this call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

Implemented in **decaf::util::AbstractCollection< E >** (p. 155), **decaf::util::AbstractSet< E >** (p. 167), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 155), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 155), **decaf::util::AbstractCollection< CompositeTask * >** (p. 155), **decaf::util::AbstractCollection< URI >** (p. 155), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 155), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 155), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 155), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 155), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 155), **decaf::util::AbstractSet< transport::TransportListener * >** (p. 167), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 167), and **decaf::util::AbstractSet< ActiveMQSession * >** (p. 167).

```
6.178.3.10  template<typename E> virtual bool decaf::util::Collection<
            E >::retainAll (const Collection< E > & collection)
            throw ( lang::exceptions::UnsupportedOperationException,
                    lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters:

collection - The **Collection** (p. 1054) whose elements are to be retained

Returns:

true if the collection changed as a result of this call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

Implemented in **decaf::util::AbstractCollection< E >** (p. 156), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 156), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 156), **decaf::util::AbstractCollection< CompositeTask * >** (p. 156), **decaf::util::AbstractCollection< URI >** (p. 156), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 156), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 156), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 156), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 156), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 156).

6.178.3.11 `template<typename E> virtual std::size_t decaf::util::Collection< E >::size () const [pure virtual]`

Returns the number of elements in this collection. If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

Returns:

the number of elements in this collection

Implemented in `decaf::util::PriorityQueue< E >` (p. 2577), `decaf::util::StlList< E >` (p. 3028), `decaf::util::StlSet< E >` (p. 3056), `decaf::util::StlList< CompositeTask * >` (p. 3028), `decaf::util::StlList< URI >` (p. 3028), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3028), `decaf::util::StlList< PrimitiveValueNode >` (p. 3028), `decaf::util::StlList< Pointer< Command > >` (p. 3028), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3028), `decaf::util::StlSet< transport::TransportListener * >` (p. 3056), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3056), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3056).

Referenced by `decaf::util::AbstractCollection< Pointer< BackupTransport > >::equals()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::isEmpty()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::toArray()`.

6.178.3.12 `template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray () const [pure virtual]`

Returns an array containing all of the elements in this collection. If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

Returns:

an array of the elements in this collection.

Implemented in `decaf::util::AbstractCollection< E >` (p. 157), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 157), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 157), `decaf::util::AbstractCollection< CompositeTask * >` (p. 157), `decaf::util::AbstractCollection< URI >` (p. 157), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 157), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 157), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 157), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 157), and `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 157).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Collection.h`

6.179 activemq::commands::Command Class Reference

#include <src/main/activemq/commands/Command.h> Inheritance diagram for activemq::commands::Command:

Public Member Functions

- virtual **~Command** ()
- virtual void **setCommandId** (int id)=0
*Sets the **Command** (p. 1063) Id of this **Message** (p. 2145).*
- virtual int **getCommandId** () const =0
*Gets the **Command** (p. 1063) Id of this **Message** (p. 2145).*
- virtual void **setResponseRequired** (const bool required)=0
*Set if this **Message** (p. 2145) requires a **Response** (p. 2781).*
- virtual bool **isResponseRequired** () const =0
*Is a **Response** (p. 2781) required for this **Command** (p. 1063).*
- virtual std::string **toString** () const =0
Returns a provider-specific string that provides information about the contents of the command.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor)=0 throw (**exceptions::ActiveMQException**)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual bool **isConnectionInfo** () const =0
- virtual bool **isConsumerInfo** () const =0
- virtual bool **isBrokerInfo** () const =0
- virtual bool **isKeepAliveInfo** () const =0
- virtual bool **isMessage** () const =0
- virtual bool **isMessageAck** () const =0
- virtual bool **isMessageDispatch** () const =0
- virtual bool **isMessageDispatchNotification** () const =0
- virtual bool **isProducerAck** () const =0
- virtual bool **isProducerInfo** () const =0
- virtual bool **isResponse** () const =0
- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

6.179.1 Constructor & Destructor Documentation

6.179.1.1 `virtual activemq::commands::Command::~~Command () [inline, virtual]`

6.179.2 Member Function Documentation

6.179.2.1 `virtual int activemq::commands::Command::getCommandId () const [pure virtual]`

Gets the **Command** (p. 1063) Id of this **Message** (p. 2145).

Returns:

Command (p. 1063) Id

Implemented in **activemq::commands::BaseCommand** (p. 652).

6.179.2.2 `virtual bool activemq::commands::Command::isBrokerInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 653), and **activemq::commands::BrokerInfo** (p. 779).

6.179.2.3 `virtual bool activemq::commands::Command::isConnectionInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 653), and **activemq::commands::ConnectionInfo** (p. 1214).

6.179.2.4 `virtual bool activemq::commands::Command::isConsumerInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 653), and **activemq::commands::ConsumerInfo** (p. 1304).

6.179.2.5 `virtual bool activemq::commands::Command::isKeepAliveInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 653), and **activemq::commands::KeepAliveInfo** (p. 1932).

6.179.2.6 `virtual bool activemq::commands::Command::isMessage () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 653), and **activemq::commands::Message** (p. 2156).

6.179.2.7 `virtual bool activemq::commands::Command::isMessageAck () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 653), and `activemq::commands::MessageAck` (p. 2191).

6.179.2.8 `virtual bool activemq::commands::Command::isMessageDispatch () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 653), and `activemq::commands::MessageDispatch` (p. 2222).

6.179.2.9 `virtual bool activemq::commands::Command::isMessageDispatchNotification () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 654), and `activemq::commands::MessageDispatchNotification` (p. 2254).

6.179.2.10 `virtual bool activemq::commands::Command::isProducerAck () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 654), and `activemq::commands::ProducerAck` (p. 2581).

6.179.2.11 `virtual bool activemq::commands::Command::isProducerInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 654), and `activemq::commands::ProducerInfo` (p. 2634).

6.179.2.12 `virtual bool activemq::commands::Command::isRemoveInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 654), and `activemq::commands::RemoveInfo` (p. 2705).

6.179.2.13 `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 654), and `activemq::commands::RemoveSubscriptionInfo` (p. 2730).

6.179.2.14 `virtual bool activemq::commands::Command::isResponse () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 654), and `activemq::commands::Response` (p. 2783).

6.179.2.15 `virtual bool activemq::commands::Command::isResponseRequired () const [pure virtual]`

Is a **Response** (p. 2781) required for this **Command** (p. 1063).

Returns:

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p. 654).

6.179.2.16 `virtual bool activemq::commands::Command::isShutdownInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 655), and **activemq::commands::ShutdownInfo** (p. 2936).

6.179.2.17 `virtual bool activemq::commands::Command::isTransactionInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 655), and **activemq::commands::TransactionInfo** (p. 3242).

6.179.2.18 `virtual bool activemq::commands::Command::isWireFormatInfo () const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 655), and **activemq::commands::WireFormatInfo** (p. 3376).

6.179.2.19 `virtual void activemq::commands::Command::setCommandId (int id) [pure virtual]`

Sets the **Command** (p. 1063) Id of this **Message** (p. 2145).

Parameters:

id **Command** (p. 1063) Id

Implemented in **activemq::commands::BaseCommand** (p. 655).

6.179.2.20 `virtual void activemq::commands::Command::setResponseRequired (const bool required) [pure virtual]`

Set if this **Message** (p. 2145) requires a **Response** (p. 2781).

Parameters:

required true if response is required

Implemented in **activemq::commands::BaseCommand** (p. 655).

6.179.2.21 virtual std::string activemq::commands::Command::toString () const [pure virtual]

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 718).

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 176), `activemq::commands::ActiveMQBytesMessage` (p. 208), `activemq::commands::ActiveMQMapMessage` (p. 320), `activemq::commands::ActiveMQMessage` (p. 344), `activemq::commands::ActiveMQObjectMessage` (p. 385), `activemq::commands::ActiveMQStreamMessage` (p. 474), `activemq::commands::ActiveMQTextMessage` (p. 577), `activemq::commands::BaseCommand` (p. 655), `activemq::commands::BrokerInfo` (p. 780), `activemq::commands::ConnectionControl` (p. 1138), `activemq::commands::ConnectionError` (p. 1162), `activemq::commands::ConnectionInfo` (p. 1215), `activemq::commands::ConsumerControl` (p. 1253), `activemq::commands::ConsumerInfo` (p. 1305), `activemq::commands::ControlCommand` (p. 1332), `activemq::commands::DataArrayResponse` (p. 1358), `activemq::commands::DataResponse` (p. 1397), `activemq::commands::DestinationInfo` (p. 1487), `activemq::commands::ExceptionResponse` (p. 1584), `activemq::commands::FlushCommand` (p. 1678), `activemq::commands::IntegerResponse` (p. 1779), `activemq::commands::KeepAliveInfo` (p. 1932), `activemq::commands::Message` (p. 2159), `activemq::commands::MessageAck` (p. 2192), `activemq::commands::MessageDispatch` (p. 2222), `activemq::commands::MessageDispatchNotification` (p. 2255), `activemq::commands::MessagePull` (p. 2349), `activemq::commands::ProducerAck` (p. 2581), `activemq::commands::ProducerInfo` (p. 2634), `activemq::commands::RemoveInfo` (p. 2705), `activemq::commands::RemoveSubscriptionInfo` (p. 2730), `activemq::commands::ReplayCommand` (p. 2754), `activemq::commands::Response` (p. 2783), `activemq::commands::SessionInfo` (p. 2879), `activemq::commands::ShutdownInfo` (p. 2936), `activemq::commands::TransactionInfo` (p. 3243), and `activemq::commands::WireFormatInfo` (p. 3378).

6.179.2.22 virtual decaf::lang::Pointer<commands::Command> activemq::commands::Command::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [pure virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implemented in `activemq::commands::BrokerError` (p. 748), `activemq::commands::BrokerInfo` (p. 781), `activemq::commands::ConnectionControl` (p. 1138), `activemq::commands::ConnectionError` (p. 1163), `activemq::commands::ConnectionInfo` (p. 1215), `activemq::commands::ConsumerControl` (p. 1254), `activemq::commands::ConsumerInfo` (p. 1306), `activemq::commands::ControlCommand` (p. 1332), `activemq::commands::DestinationInfo` (p. 1487), `activemq::commands::FlushCommand` (p. 1678), `activemq::commands::KeepAliveInfo` (p. 1932), `activemq::commands::Message` (p. 2160), `activemq::commands::MessageAck` (p. 2192), `activemq::commands::MessageDispatch`

(p. 2223), **activemq::commands::MessageDispatchNotification**
(p. 2255), **activemq::commands::MessagePull** (p. 2349), **ac-**
tivemq::commands::ProducerAck (p. 2582), **activemq::commands::ProducerInfo**
(p. 2635), **activemq::commands::RemoveInfo** (p. 2706), **ac-**
tivemq::commands::RemoveSubscriptionInfo (p. 2730), **ac-**
tivemq::commands::ReplayCommand (p. 2754), **activemq::commands::Response**
(p. 2784), **activemq::commands::SessionInfo** (p. 2879), **ac-**
tivemq::commands::ShutdownInfo (p. 2936), **activemq::commands::TransactionInfo**
(p. 3243), and **activemq::commands::WireFormatInfo** (p. 3378).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Command.h`

6.180 activemq::state::CommandVisitor Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

#include <src/main/activemq/state/CommandVisitor.h> Inheritance diagram for activemq::state::CommandVisitor:

Public Member Functions

- virtual `~CommandVisitor ()`
- virtual `decaf::lang::Pointer< commands::Command > processTransactionInfo (commands::TransactionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveInfo (commands::RemoveInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processConnectionInfo (commands::ConnectionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processSessionInfo (commands::SessionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processProducerInfo (commands::ProducerInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processConsumerInfo (commands::ConsumerInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConsumer (commands::ConsumerId *id)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processDestinationInfo (commands::DestinationInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveDestination (commands::DestinationInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processMessage (commands::Message *send)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processMessageAck (commands::MessageAck *ack)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processMessagePull (commands::MessagePull *pull)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processBeginTransaction (commands::TransactionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processPrepareTransaction (commands::TransactionInfo *info)=0` throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** *error)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** *response)=0 throw (exceptions::ActiveMQException)

6.180.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client. The Commands themselves implement a **visit** method that is called with an instance of this interface and each one then call the appropriate **processXXX** method.

Since:

3.0

6.180.2 Constructor & Destructor Documentation

6.180.2.1 virtual `activemq::state::CommandVisitor::~~CommandVisitor ()` [inline, virtual]

6.180.3 Member Function Documentation

6.180.3.1 virtual `decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBeginTransaction (commands::TransactionInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1246).

6.180.3.2 virtual `decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBrokerError (commands::BrokerError * error) throw (exceptions::ActiveMQException)` [pure virtual]

6.180.3.3 virtual `decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBrokerInfo (commands::BrokerInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

6.180.3.4 virtual `decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processCommitTransactionOnePhase (commands::TransactionInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1246).

6.180.3.5 virtual `decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processCommitTransactionTwoPhase (commands::TransactionInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1246).

- 6.180.3.6** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processConnectionControl
 (commands::ConnectionControl * *control*) throw (
 exceptions::ActiveMQException) [pure virtual]
- 6.180.3.7** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processConnectionError
 (commands::ConnectionError * *error*) throw (
 exceptions::ActiveMQException) [pure virtual]
- 6.180.3.8** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processConnectionInfo
 (commands::ConnectionInfo * *info*) throw (
 exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

- 6.180.3.9** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processConsumerControl
 (commands::ConsumerControl * *control*) throw (
 exceptions::ActiveMQException) [pure virtual]
- 6.180.3.10** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processConsumerInfo
 (commands::ConsumerInfo * *info*) throw (
 exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

- 6.180.3.11** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processControlCommand
 (commands::ControlCommand * *command*) throw (
 exceptions::ActiveMQException) [pure virtual]
- 6.180.3.12** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processDestinationInfo
 (commands::DestinationInfo * *info*) throw (
 exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

- 6.180.3.13** virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processEndTransaction
 (commands::TransactionInfo * *info*) throw (
 exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

- 6.180.3.14** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processFlushCommand
(commands::FlushCommand * *command*) throw (
exceptions::ActiveMQException) [pure virtual]
- 6.180.3.15** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processForgetTransaction
(commands::TransactionInfo * *info*) throw (
exceptions::ActiveMQException) [pure virtual]
- 6.180.3.16** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processKeepAliveInfo
(commands::KeepAliveInfo * *info*) throw (
exceptions::ActiveMQException) [pure virtual]
- 6.180.3.17** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessage (commands::Message
* *send*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

- 6.180.3.18** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageAck
(commands::MessageAck * *ack*) throw (exceptions::ActiveMQException
) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

- 6.180.3.19** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageDispatch
(commands::MessageDispatch * *dispatch*) throw (
exceptions::ActiveMQException) [pure virtual]
- 6.180.3.20** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageDispatchNotification
(commands::MessageDispatchNotification * *notification*) throw (
exceptions::ActiveMQException) [pure virtual]
- 6.180.3.21** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessagePull
(commands::MessagePull * *pull*) throw (exceptions::ActiveMQException
) [pure virtual]
- 6.180.3.22** virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processPrepareTransaction
(commands::TransactionInfo * *info*) throw (
exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1247).

6.180.3.23 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerAck
(commands::ProducerAck * ack) throw (exceptions::ActiveMQException
)` [pure virtual]

6.180.3.24 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerInfo (com-
mands::ProducerInfo * info) throw (exceptions::ActiveMQException)
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.25 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRecoverTransactions
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

6.180.3.26 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConnection
(commands::ConnectionId * id) throw (exceptions::ActiveMQException
)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.27 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConsumer
(commands::ConsumerId * id) throw (exceptions::ActiveMQException
)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.28 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveDestination
(commands::DestinationInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.29 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveInfo
(commands::RemoveInfo * info) throw (exceptions::ActiveMQException
)` [pure virtual]

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1080).

6.180.3.30 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveProducer
(commands::ProducerId * id) throw (exceptions::ActiveMQException)
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.31 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processRemoveSession
 (commands::SessionId * *id*) throw (exceptions::ActiveMQException)
 [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.32 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processRemoveSubscriptionInfo
 (commands::RemoveSubscriptionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.33 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processReplayCommand
 (commands::ReplayCommand * *replay*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.34 virtual decaf::lang::Pointer<commands::Command> ac-
 tivemq::state::CommandVisitor::processResponse (commands::Response
 * *response*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.35 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processRollbackTransaction
 (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1248).

6.180.3.36 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processSessionInfo
 (commands::SessionInfo * *info*) throw (exceptions::ActiveMQException)
 [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1249).

6.180.3.37 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processShutdownInfo (com-
 mands::ShutdownInfo * *info*) throw (exceptions::ActiveMQException)
 [pure virtual]

6.180.3.38 virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processTransactionInfo
 (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1081).

6.180.3.39 `virtual decaf::lang::Pointer<commands::Command>
 activemq::state::CommandVisitor::processWireFormat
 (commands::WireFormatInfo * info) throw (
 exceptions::ActiveMQException) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitor.h`

6.181 activemq::state::CommandVisitorAdapter Class Reference

Default Implementation of a **CommandVisitor** (p.1069) that returns NULL for all calls.

#include <src/main/activemq/state/CommandVisitorAdapter.h> Inheritance diagram for activemq::state::CommandVisitorAdapter:

Public Member Functions

- virtual **~CommandVisitorAdapter** ()
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConnection** (commands::ConnectionId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSession** (commands::SessionId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveProducer** (commands::ProducerId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (commands::ConsumerId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (commands::DestinationInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (commands::DestinationInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (commands::Message *send AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (commands::MessageAck *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (commands::MessagePull *pull AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processBrokerError** (**commands::BrokerError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processReplayCommand** (**commands::ReplayCommand** *replay AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processResponse** (**commands::Response** *response AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionInfo** (**commands::ConnectionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processSessionInfo** (**commands::SessionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processProducerInfo** (**commands::ProducerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConsumerInfo** (**commands::ConsumerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processTransactionInfo** (**commands::TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveInfo** (**commands::RemoveInfo** *info) throw (exceptions::ActiveMQException)

6.181.1 Detailed Description

Default Implementation of a **CommandVisitor** (p.1069) that returns NULL for all calls.

Since:

3.0

6.181.2 Constructor & Destructor Documentation

- 6.181.2.1 virtual
activemq::state::CommandVisitorAdapter::~~CommandVisitorAdapter ()
[inline, virtual]

6.181.3 Member Function Documentation

- 6.181.3.1 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBeginTransaction
(commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.2 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerError
(commands::BrokerError *error *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.3 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerInfo
(commands::BrokerInfo *info *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.4 virtual decaf::lang::Pointer<commands::Command> ac-
tivemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase
(commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.5 virtual decaf::lang::Pointer<commands::Command> ac-
tivemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase
(commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.6 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionControl
(commands::ConnectionControl *control *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.7 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionError
(commands::ConnectionError *error *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.8 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionInfo
(commands::ConnectionInfo *info *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.9 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerControl
(commands::ConsumerControl *control *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.10 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerInfo
(commands::ConsumerInfo *info *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]
- 6.181.3.11 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processControlCommand
(commands::ControlCommand *command *AMQCPP_UNUSED*) throw (
exceptions::ActiveMQException) [inline, virtual]

References `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

- 6.181.3.30 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRemoveProducer`
`(commands::ProducerId *id AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.31 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRemoveSession`
`(commands::SessionId *id AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.32 `virtual decaf::lang::Pointer<commands::Command>` `ac-`
`tivemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo`
`(commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED)`
`throw (exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.33 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processReplayCommand`
`(commands::ReplayCommand *replay AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.34 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processResponse`
`(commands::Response *response AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.35 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRollbackTransaction`
`(commands::TransactionInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.36 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processSessionInfo`
`(commands::SessionInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.37 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processShutdownInfo`
`(commands::ShutdownInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.38 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processTransactionInfo`
`(commands::TransactionInfo * info) throw (`
`exceptions::ActiveMQException) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1075).

References `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_`

```

COMMITONEPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
COMMITTWO PHASE,      activemq::core::ActiveMQConstants::TRANSACTION_-
STATE_END,            activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
FORGET,               activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE,
activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER,      and
activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK.

```

```

6.181.3.39  virtual decaf::lang::Pointer<commands::Command>
               activemq::state::CommandVisitorAdapter::processWireFormat
               (commands::WireFormatInfo *info AMQCPP_UNUSED) throw (
               exceptions::ActiveMQException ) [inline, virtual]

```

The documentation for this class was generated from the following file:

- src/main/activemq/state/**CommandVisitorAdapter.h**

6.182 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

Public Member Functions

- virtual **~Comparable** ()
- virtual int **compareTo** (const T &value) const =0
Compares this object with the specified object for order.
- virtual bool **equals** (const T &value) const =0
- virtual bool **operator==** (const T &value) const =0
Compares equality between this object and the one passed.
- virtual bool **operator<** (const T &value) const =0
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.182.1 Detailed Description

```
template<typename T> class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 `template<typename T> virtual decaf::lang::Comparable< T >::~~Comparable () [inline, virtual]`

6.182.3 Member Function Documentation

6.182.3.1 `template<typename T> virtual int decaf::lang::Comparable< T >::compareTo (const T & value) const [pure virtual]`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all x and y. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementer must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all `z`.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p.1083) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in `decaf::lang::Boolean` (p.734), `decaf::lang::Boolean` (p.733), `decaf::lang::Byte` (p.842), `decaf::lang::Byte` (p.842), `decaf::lang::Character` (p.984), `decaf::lang::Character` (p.984), `decaf::lang::Double` (p.1540), `decaf::lang::Double` (p.1540), `decaf::lang::Float` (p.1650), `decaf::lang::Float` (p.1650), `decaf::lang::Integer` (p.1766), `decaf::lang::Integer` (p.1765), `decaf::lang::Long` (p.2061), `decaf::lang::Long` (p.2060), `decaf::lang::Short` (p.2908), and `decaf::lang::Short` (p.2908).

6.182.3.2 `template<typename T> virtual bool decaf::lang::Comparable< T >::equals (const T & value) const` [pure virtual]

Returns:

true if this value is considered equal to the passed value.

Implemented in `decaf::lang::Boolean` (p.734), `decaf::lang::Boolean` (p.734), `decaf::lang::Byte` (p.844), `decaf::lang::Byte` (p.843), `decaf::lang::Character` (p.985), `decaf::lang::Character` (p.985), `decaf::lang::Double` (p.1542), `decaf::lang::Double` (p.1542), `decaf::lang::Float` (p.1651), `decaf::lang::Float` (p.1651), `decaf::lang::Integer` (p.1767), `decaf::lang::Integer` (p.1767), `decaf::lang::Long` (p.2062), `decaf::lang::Long` (p.2061), `decaf::lang::Short` (p.2909), and `decaf::lang::Short` (p.2909).

6.182.3.3 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< (const T & value) const` [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p.735), `decaf::lang::Boolean` (p.734), `decaf::lang::Byte` (p.845), `decaf::lang::Byte` (p.844), `decaf::lang::Character` (p.987), `decaf::lang::Character` (p.987), `decaf::lang::Double` (p.1544), `decaf::lang::Double` (p.1544),

decaf::lang::Float (p. 1654), **decaf::lang::Float** (p. 1654), **decaf::lang::Integer** (p. 1769), **decaf::lang::Integer** (p. 1769), **decaf::lang::Long** (p. 2064), **decaf::lang::Long** (p. 2064), **decaf::lang::Short** (p. 2910), and **decaf::lang::Short** (p. 2910).

6.182.3.4 `template<typename T> virtual bool decaf::lang::Comparable< T
>::operator==(const T & value) const [pure virtual]`

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 735), **decaf::lang::Boolean** (p. 735), **decaf::lang::Byte** (p. 845), **decaf::lang::Byte** (p. 845), **decaf::lang::Character** (p. 988), **decaf::lang::Character** (p. 987), **decaf::lang::Double** (p. 1544), **decaf::lang::Double** (p. 1544), **decaf::lang::Float** (p. 1654), **decaf::lang::Float** (p. 1654), **decaf::lang::Integer** (p. 1770), **decaf::lang::Integer** (p. 1769), **decaf::lang::Long** (p. 2065), **decaf::lang::Long** (p. 2064), **decaf::lang::Short** (p. 2911), and **decaf::lang::Short** (p. 2911).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Comparable.h`

6.183 decaf::util::Comparator< T > Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

Public Member Functions

- virtual `~Comparator()`
- virtual `bool operator()(const T &left, const T &right) const =0`
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1086) to be passed to an STL **Map** (p. 2094) for use as the sorting criteria.*
- virtual `int compare(const T &o1, const T &o2) const =0`
Compares its two arguments for order.

6.183.1 Detailed Description

```
template<typename T> class decaf::util::Comparator< T >
```

A comparison function, which imposes a total ordering on some collection of objects. Comparators can be passed to a sort method (such as `Collections.sort()`) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 1086) *c* on a set of elements *S* is said to be consistent with equals if and only if (`compare(e1, e2) == 0`) has the same boolean value as (`e1 == e2`) for every *e1* and *e2* in *S*.

6.183.2 Constructor & Destructor Documentation

6.183.2.1 `template<typename T> virtual decaf::util::Comparator< T >::~Comparator()` [`inline`, `virtual`]

6.183.3 Member Function Documentation

6.183.3.1 `template<typename T> virtual int decaf::util::Comparator< T >::compare(const T & o1, const T & o2) const` [`pure virtual`]

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn(compare(x, y)) == -sgn(compare(y, x))` for all *x* and *y*. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0)) implies compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all *z*.

It is generally the case, but not strictly required that `(compare(x, y)==0) == (x == y)`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters:

- o1* - the first object to be compared
- o2* - the second object to be compared

Returns:

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in `decaf::util::comparators::Less< E >` (p.1981).

6.183.3.2 `template<typename T> virtual bool decaf::util::Comparator< T >::operator() (const T & left, const T & right) const` [pure virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p.1086) to be passed to an STL **Map** (p.2094) for use as the sorting criteria.

Parameters:

- left* - the Left hand side operand.
- right* - the Right hand side operand.

Returns:

true if the value of left is less than the value of right.

Implemented in `decaf::util::comparators::Less< E >` (p.1982).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.184 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

Public Member Functions

- **CompositeData** ()
- virtual **~CompositeData** ()
- **StlList< URI > & getComponents** ()
- const **StlList< URI > & getComponents** () const
- void **setComponents** (const **StlList< URI > &components**)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties & getParameters** () const
- void **setParameters** (const **Properties ¶meters**)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw (decaf::net::URISyntaxException)

6.184.1 Detailed Description

Represents a Composite URI.

Since:

3.0

6.184.2 Constructor & Destructor Documentation

6.184.2.1 `activemq::util::CompositeData::CompositeData ()`

6.184.2.2 `virtual activemq::util::CompositeData::~~CompositeData ()` [virtual]

6.184.3 Member Function Documentation

6.184.3.1 `const StlList<URI>& activemq::util::CompositeData::getComponents ()`
const [inline]

6.184.3.2 `StlList<URI>& activemq::util::CompositeData::getComponents ()`
[inline]

6.184.3.3 `std::string activemq::util::CompositeData::getFragment ()` const
[inline]

6.184.3.4 `std::string activemq::util::CompositeData::getHost ()` const [inline]

6.184.3.5 `const Properties& activemq::util::CompositeData::getParameters ()`
const [inline]

6.184.3.6 `std::string activemq::util::CompositeData::getPath ()` const [inline]

6.184.3.7 `std::string activemq::util::CompositeData::getScheme ()` const [inline]

6.184.3.8 `void activemq::util::CompositeData::setComponents (const StlList< URI`
> & *components*) [inline]

6.184.3.9 `void activemq::util::CompositeData::setFragment (const std::string &`
fragment) [inline]

6.184.3.10 `void activemq::util::CompositeData::setHost (const std::string & host)`
[inline]

6.184.3.11 `void activemq::util::CompositeData::setParameters (const Properties &`
parameters) [inline]

6.184.3.12 `void activemq::util::CompositeData::setPath (const std::string & path)`
[inline]

6.184.3.13 `void activemq::util::CompositeData::setScheme (const std::string &`
scheme) [inline]

6.184.3.14 `URI activemq::util::CompositeData::toURI ()` const throw (
`decaf::net::URISyntaxException`)

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

6.185 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a `CompositeTaskRunner` (p.1092).

#include <src/main/activemq/threads/CompositeTask.h> Inheritance diagram for `activemq::threads::CompositeTask`:

Public Member Functions

- virtual `~CompositeTask ()`
- virtual bool `isPending () const =0`

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3148) in the `CompositeTaskRunner`'s list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.*

6.185.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a `CompositeTaskRunner` (p.1092).

Since:

3.0

6.185.2 Constructor & Destructor Documentation

6.185.2.1 virtual `activemq::threads::CompositeTask::~~CompositeTask ()` [inline, virtual]

6.185.3 Member Function Documentation

6.185.3.1 virtual bool `activemq::threads::CompositeTask::isPending () const` [pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3148) in the `CompositeTaskRunner`'s list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

Since:

3.0

Implemented in `activemq::transport::failover::BackupTransportPool` (p.649), `activemq::transport::failover::CloseTransportsTask` (p.1022), and `activemq::transport::failover::FailoverTransport` (p.1618).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

6.186 activemq::threads::CompositeTaskRunner Class Reference

A **Task** (p. 3148) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

#include <src/main/activemq/threads/CompositeTaskRunner.h> Inheritance diagram for activemq::threads::CompositeTaskRunner:

Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (CompositeTask *task)
*Adds a new **CompositeTask** (p. 1090) to the Set of Tasks that this class manages.*
- void **removeTask** (CompositeTask *task)
*Removes a **CompositeTask** (p. 1090) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()
Shutdown once the task has finished and the TaskRunner's thread has exited.
- virtual void **wakeup** ()
*Signal the **TaskRunner** (p. 3150) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3148) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()
Run method - called by the Thread class in the context of the thread.
- virtual bool **iterate** ()
Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.186.1 Detailed Description

A **Task** (p. 3148) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Since:

3.0

6.186.2 Constructor & Destructor Documentation

6.186.2.1 `activemq::threads::CompositeTaskRunner::CompositeTaskRunner ()`

6.186.2.2 `virtual
activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner ()
[virtual]`

6.186.3 Member Function Documentation

6.186.3.1 `void activemq::threads::CompositeTaskRunner::addTask (CompositeTask
* task)`

Adds a new **CompositeTask** (p. 1090) to the Set of Tasks that this class manages.

Parameters:

task - Pointer to a **CompositeTask** (p. 1090) instance.

6.186.3.2 `virtual bool activemq::threads::CompositeTaskRunner::iterate ()
[protected, virtual]`

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns:

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implements **activemq::threads::Task** (p. 3148).

6.186.3.3 `void activemq::threads::CompositeTaskRunner::removeTask
(CompositeTask * task)`

Removes a **CompositeTask** (p. 1090) that was added previously.

Parameters:

task - Pointer to a **CompositeTask** (p. 1090) instance.

6.186.3.4 `virtual void activemq::threads::CompositeTaskRunner::run ()
[protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2816).

6.186.3.5 `virtual void activemq::threads::CompositeTaskRunner::shutdown ()
[virtual]`

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3150).

6.186.3.6 **virtual void activemq::threads::CompositeTaskRunner::shutdown**
 (unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters:

timeout - Time in Milliseconds to wait for the task to stop.

Implements **activemq::threads::TaskRunner** (p. 3150).

6.186.3.7 **virtual void activemq::threads::CompositeTaskRunner::wakeup ()**
 [virtual]

Signal the **TaskRunner** (p. 3150) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3148) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3151).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTaskRunner.h`

6.187 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 3273) is a **Transport** (p. 3273) implementation that is composed of several Transports.

#include <src/main/activemq/transport/CompositeTransport.h> Inheritance diagram for activemq::transport::CompositeTransport:

Public Member Functions

- virtual **~CompositeTransport** ()
- virtual void **addURI** (const **List**< **URI** > &uris)=0
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3273) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)=0
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3273) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3273) should result in that **Transport** (p. 3273) being disposed of.*

6.187.1 Detailed Description

A Composite **Transport** (p. 3273) is a **Transport** (p. 3273) implementation that is composed of several Transports. The composition could be such that only one **Transport** (p. 3273) exists for each URI that is composed or there could be many active Transports working at once.

Since:

3.0

6.187.2 Constructor & Destructor Documentation

- 6.187.2.1** virtual **activemq::transport::CompositeTransport::~CompositeTransport** () [inline, virtual]

6.187.3 Member Function Documentation

- 6.187.3.1** virtual void **activemq::transport::CompositeTransport::addURI** (const **List**< **URI** > & *uris*) [pure virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3273) is a composite of.

Parameters:

uris The new URI set to add to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1615).

6.187.3.2 virtual void activemq::transport::CompositeTransport::removeURI (const List< URI > & *uris*) [pure virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3273) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3273) should result in that **Transport** (p. 3273) being disposed of.

Parameters:

uris The new URI set to remove to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1619).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

6.188 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 2094) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2094) interface.

#include <src/main/decaf/util/concurrent/ConcurrentMap.h> Inheritance diagram for decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >:

Public Member Functions

- virtual **~ConcurrentMap** ()
- virtual bool **putIfAbsent** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

If the specified key is not already associated with a value, associate it with the given value.

- virtual bool **remove** (const K &key, const V &value)=0

Remove entry for key only if currently mapped to given value.

- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0

Replace entry for key only if currently mapped to given value.

- virtual V **replace** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)

Replace entry for key only if currently mapped to some value.

6.188.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR> class
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >
```

Interface for a **Map** (p. 2094) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2094) interface.

Since:

1.0

6.188.2 Constructor & Destructor Documentation

6.188.2.1 `template<typename K, typename V, typename COMPARATOR>
virtual decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
>::~ConcurrentMap () [inline, virtual]`

6.188.3 Member Function Documentation

6.188.3.1 `template<typename K, typename V, typename COMPARATOR>
virtual bool decaf::util::concurrent::ConcurrentMap< K, V,
COMPARATOR >::putIfAbsent (const K & key, const V & value)
throw (decaf::lang::exceptions::UnsupportedOperationException) [pure
virtual]`

If the specified key is not already associated with a value, associate it with the given value. This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key The key to map the value to.

value The value to map to the given key.

Returns:

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions:

UnsupportedOperationException if the put operation is not supported by this map

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1112), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1112).

6.188.3.2 `template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::remove (const K & key, const V & value) [pure virtual]`

Remove entry for key only if currently mapped to given value. Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value associated with the specified key.

Returns:

true if the value was removed, false otherwise

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1112), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1112), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1112).

6.188.3.3 `template<typename K, typename V, typename COMPARATOR> virtual V decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & key, const V & value) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Replace entry for key only if currently mapped to some value. Acts as

```
if( ( map.containsKey( key ) ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns:

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

Exceptions:

NoSuchElementException if there was no previous mapping.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1113), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1113).

6.188.3.4 `template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue) [pure virtual]`

Replace entry for key only if currently mapped to given value. Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns:

true if the value was replaced

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p.1114), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1114).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentMap.h`

6.189 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h> Inheritance diagram for `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >`:

Public Member Functions

- **ConcurrentStlMap** ()
Default constructor - does nothing.
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **ConcurrentStlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
Copies the content of the source map into this map.
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
Exceptions:
***UnsupportedOperationException** if this map is unmodifiable.*
- virtual bool **containsKey** (const K &key) const
Indicates whether or this map contains a value for the given key.
Parameters:
key The key to look up.
Returns:
true if this map contains the value, otherwise false.
- virtual bool **containsValue** (const V &value) const

Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const

Returns:

*if the **Map** (p. 2094) contains any element or not, TRUE or FALSE*

- virtual std::size_t **size** () const

Returns:

The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 2094).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.*

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2094).*

- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 2094).*

*If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.*

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2094).*

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

Sets the value for the specified key.

Parameters:

key The target key.

value The value to be set.

Exceptions:

***UnsupportedOperationException** if this map is unmodifiable.*

- virtual void **putAll** (const **ConcurrentStlMap**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

*Stores a copy of the Mappings contained in the other **Map** (p. 2094) in this one.*

Parameters:

*other A **Map** (p. 2094) instance whose elements are to all be inserted in this **Map** (p. 2094).*

Exceptions:

UnsupportedOperationException *If the implementing class does not support the putAll operation.*

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException *if this key is not in the **Map** (p. 2094).*

UnsupportedOperationException *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 2905) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1833), **Set.remove** (p. 155), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.*

Returns:

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

Returns:

the entire set of values in this map as a std::vector.

- bool **putIfAbsent** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

If the specified key is not already associated with a value, associate it with the given value.

- bool **remove** (const K &key, const V &value)

Remove entry for key only if currently mapped to given value.

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

Replace entry for key only if currently mapped to given value.

- `V replace (const K &key, const V &value) throw (decaf::lang::exceptions::NoSuchElementException)`
Replace entry for key only if currently mapped to some value.
- `virtual void lock () throw (decaf::lang::exceptions::RuntimeException)`
Locks the object.
- `virtual bool tryLock () throw (decaf::lang::exceptions::RuntimeException)`
*Attempts to **Lock** (p. 2023) the object, if the lock is already held by another thread than this method returns false.*
- `virtual void unlock () throw (decaf::lang::exceptions::RuntimeException)`
Unlocks the object.
- `virtual void wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)`
Signals a waiter on this object that it can now wake up and continue.
- `virtual void notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)`
Signals the waiters on this object that it can now wake up and continue.

6.189.1 Detailed Description

`template<typename K, typename V, typename COMPARATOR = std::less<K>>`
`class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >`

Map (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`. This version of **Map** (p. 2094) extends the **ConcurrentMap** (p. 1097) interface and implements all the methods defined in that interface. Unlike a Java `ConcurrentHashMap` this implementations synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java `HashTable`.

Since:

1.0

6.189.2 Constructor & Destructor Documentation

6.189.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap () [inline]`

Default constructor - does nothing.

6.189.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap (const ConcurrentStlMap< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.189.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap (const Map< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.189.2.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::~~ConcurrentStlMap () [inline, virtual]`

6.189.3 Member Function Documentation

6.189.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::clear () throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all keys and values from this map.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2095).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy().

6.189.3.2 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::containsKey (const K & key) const [inline,
 virtual]`

Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2096).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::remove(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace().

6.189.3.3 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::containsValue (const V & value) const
 [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2097).

6.189.3.4 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR >
 & source) [inline, virtual]`

Copies the content of the source map into this map. Erases all existing data in this map.

Parameters:

source The source object to copy from.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2098).

6.189.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const ConcurrentStlMap< K, V, COMPARATOR > & source) [inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap()`.

6.189.3.6 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters:

source - **Map** (p. 2094) to compare to this one.

Returns:

true if the **Map** (p. 2094) passed is equal in value to this one.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2098).

6.189.3.7 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const ConcurrentStlMap< K, V, COMPARATOR > & source) const [inline, virtual]`

6.189.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & key) const throw (lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2094).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2094).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2098).

6.189.3.9 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual V& decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::get (const K & key) throw (
 lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2094).

If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2094).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2099).

6.189.3.10 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::isEmpty () const [inline, virtual]`

Returns:

if the **Map** (p. 2094) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2100).

6.189.3.11 `template<typename K, typename V, typename
 COMPARATOR = std::less<K>> virtual std::vector<K>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::keySet () const [inline, virtual]`

Returns a **Set** (p. 2905) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1833), **Set.remove** (p. 155), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns:

the entire set of keys in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2101).

```
6.189.3.12  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::lock () throw ( decaf::lang::exceptions::RuntimeException ) [inline,
             virtual]
```

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

```
6.189.3.13  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::notify () throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
             virtual]
```

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

```
6.189.3.14  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::notifyAll () throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
             virtual]
```

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3125).

6.189.3.15 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::put (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Sets the value for the specified key.

Parameters:

key The target key.
value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2102).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.189.3.16 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2094) in this one.

Parameters:

other A **Map** (p. 2094) instance whose elements are to all be inserted in this **Map** (p. 2094).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2102).

6.189.3.17 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putAll (const ConcurrentStlMap< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.189.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putIfAbsent (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

If the specified key is not already associated with a value, associate it with the given value. This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key The key to map the value to.

value The value to map to the given key.

Returns:

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions:

UnsupportedOperationException if the put operation is not supported by this map

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p.1098).

6.189.3.19 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key, const V & value) [inline, virtual]`

Remove entry for key only if currently mapped to given value. Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value associated with the specified key.

Returns:

true if the value was removed, false otherwise

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1099).

6.189.3.20 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 2094).

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2103).

6.189.3.21 `template<typename K, typename V, typename COMPARATOR = std::less<K>> V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Replace entry for key only if currently mapped to some value. Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns:

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

Exceptions:

NoSuchElementException if there was no previous mapping.

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1099).

6.189.3.22 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue)` [inline, virtual]

Replace entry for key only if currently mapped to given value. Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns:

true if the value was replaced

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1100).

6.189.3.23 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::size () const` [inline, virtual]

Returns:

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2104).

6.189.3.24 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::tryLock () throw (
 decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to **Lock** (p.2023) the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p.3126).

6.189.3.25 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::unlock () throw (
 decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p.3127).

6.189.3.26 `template<typename K, typename V, typename
 COMPARATOR = std::less<K>> virtual std::vector<V>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::values () const [inline, virtual]`

Returns:

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.2105).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values()`.

6.189.3.27 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::wait (long long milliseconds, int
nanos) throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalArgumentException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is

similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

```
6.189.3.28  template<typename K, typename V, typename COMPARATOR =
             std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
             K, V, COMPARATOR >::wait (long long milliseconds)
             throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

```
6.189.3.29  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::wait () throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentStlMap.h`

6.190 decaf::util::concurrent::locks::Condition Class Reference

Condition (p. 1118) factors out the **Mutex** (p. 2385) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2017) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

Public Member Functions

- virtual **~Condition** ()
- virtual void **await** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted.
- virtual void **awaitUninterruptibly** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signalled.
- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.
- virtual bool **awaitUntil** (const **Date** &deadline)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
- virtual void **signal** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Wakes up one waiting thread.
- virtual void **signalAll** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Wakes up all waiting threads.

6.190.1 Detailed Description

Condition (p. 1118) factors out the **Mutex** (p. 2385) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2017) implementations. Where a **Lock** (p. 2017) replaces the use of synchronized statements, a **Condition** (p. 1118) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 1118) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 1118) instance for a particular **Lock** (p. 2017) instance use its `newCondition()` method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 1118) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 1120); items[putptr] = x; if ( ++putptr == 100 ) putptr = 0; ++count; notEmpty->signal() (p. 1123); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 1120); Object x = items[takeptr]; if ( ++takeptr == 100 ) takeptr = 0; --count; notFull->signal() (p. 1123); return x; } catch(...) { lock->unlock(); } }
```

(The `ArrayBlockingQueue` class provides this functionality, so there is no reason to implement this sample usage class.)

Implementation Considerations

When waiting upon a **Condition** (p. 1118), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a **Condition** (p. 1118) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since:

1.0

6.190.2 Constructor & Destructor Documentation

6.190.2.1 `virtual decaf::util::concurrent::locks::Condition::~~Condition () [inline, virtual]`

6.190.3 Member Function Documentation

6.190.3.1 `virtual bool decaf::util::concurrent::locks::Condition::await (long long time, const TimeUnit & unit) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses. This method is behaviorally equivalent to:

```
awaitNanos(unit.toNanos(time)) > 0
```

Parameters:

time - the maximum time to wait

unit - the time unit of the time argument

Returns:

false if the waiting time detectably elapsed before return from the method, else true

Exceptions:

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1118).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.190.3.2 `virtual void decaf::util::concurrent::locks::Condition::await () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted. The lock associated with this **Condition** (p. 1118) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes the **signal()** (p. 1123) method for this **Condition** (p. 1118) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p. 1123) method for this **Condition** (p. 1118); or * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting and interruption of thread suspension is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p.1118) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as IllegalMonitorStateException) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Exceptions:

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1118).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.190.3.3 `virtual long long decaf::util::concurrent::locks::Condition::awaitNanos (long long nanosTimeout) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses. The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

* Some other thread invokes the **signal()** (p.1123) method for this **Condition** (p.1118) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p.1123) method for this **Condition** (p.1118); or * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or * The specified waiting time elapses; or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting and interruption of thread suspension is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied `nanosTimeout` value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```
synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanosTimeout =
unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout > 0) nanosTimeout =
theCondition->awaitNanos(nanosTimeout); else return false; } // ... }
```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p.1118) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Parameters:

nanosTimeout - the maximum time to wait, in nanoseconds

Returns:

an estimate of the `nanosTimeout` value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

Exceptions:

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1118).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.190.3.4 virtual void deca::util::concurrent::locks::Condition::awaitUninterruptibly() **throw (deca::lang::exceptions::RuntimeException, deca::lang::exceptions::IllegalMonitorStateException)** [pure virtual]

Causes the current thread to wait until it is signalled. The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes the **signal()** (p.1123) method for this **Condition** (p.1118) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p.1123) method for this **Condition** (p.1118); or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p.1118) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

Exceptions:

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1118).

IllegalMonitorStateException if the caller is not the lock owner.

6.190.3.5 `virtual bool decaf::util::concurrent::locks::Condition::awaitUntil (const Date & deadline) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)` [pure virtual]

6.190.3.6 `virtual void decaf::util::concurrent::locks::Condition::signal () throw (decaf::lang::exceptions::RuntimeException)` [pure virtual]

Wakes up one waiting thread. If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

Exceptions:

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1118).

6.190.3.7 `virtual void decaf::util::concurrent::locks::Condition::signalAll () throw (decaf::lang::exceptions::RuntimeException)` [pure virtual]

Wakes up all waiting threads. If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

Exceptions:

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1118).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Condition.h`

6.191 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h>
```

Public Member Functions

- **ConditionHandle** ()
- **~ConditionHandle** ()
- **ConditionHandle** ()
- **~ConditionHandle** ()

Data Fields

- pthread_cond_t **condition**
- MutexHandle * **mutex**
- HANDLE **semaphore**
- CRITICAL_SECTION **criticalSection**
- volatile unsigned int **numWaiting**
- volatile unsigned int **numWake**
- volatile unsigned int **generation**

6.191.1 Constructor & Destructor Documentation

6.191.1.1 **decaf::util::concurrent::ConditionHandle::ConditionHandle** () [inline]

6.191.1.2 **decaf::util::concurrent::ConditionHandle::~~ConditionHandle** () [inline]

6.191.1.3 **decaf::util::concurrent::ConditionHandle::ConditionHandle** () [inline]

6.191.1.4 **decaf::util::concurrent::ConditionHandle::~~ConditionHandle** () [inline]

6.191.2 Field Documentation

6.191.2.1 pthread_cond_t **decaf::util::concurrent::ConditionHandle::condition**

6.191.2.2 CRITICAL_SECTION **decaf::util::concurrent::ConditionHandle::criticalSection**

6.191.2.3 volatile unsigned int **decaf::util::concurrent::ConditionHandle::generation**

6.191.2.4 MutexHandle * **decaf::util::concurrent::ConditionHandle::mutex**

6.191.2.5 volatile unsigned int **decaf::util::concurrent::ConditionHandle::numWaiting**

6.191.2.6 volatile unsigned int **decaf::util::concurrent::ConditionHandle::numWake**

6.191.2.7 HANDLE **decaf::util::concurrent::ConditionHandle::semaphore**

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h
- src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h

6.192 decaf::internal::util::concurrent::ConditionImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/ConditionImpl.h>
```

Static Public Member Functions

- static **decaf::util::concurrent::ConditionHandle** * **create** (**decaf::util::concurrent::MutexHandle** *mutex)
Creates the Condition object and attaches it to the given MutexHandle.
- static void **destroy** (**decaf::util::concurrent::ConditionHandle** *handle)
Destroy a previously create Condition instance.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition)
Waits for the condition to be signaled.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition, long long mills, long long nanos)
Waits for the condition to be signaled or for the time specified to ellapse.
- static void **notify** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals one Thread that is waiting on this condition to wake up.
- static void **notifyAll** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals all Threads that is waiting on this condition to wake up.

6.192.1 Member Function Documentation

6.192.1.1 static **decaf::util::concurrent::ConditionHandle***
decaf::internal::util::concurrent::ConditionImpl::create
(**decaf::util::concurrent::MutexHandle** * *mutex*) [static]

Creates the Condition object and attaches it to the given MutexHandle.

Parameters:

mutex the Mutex handle that this Condition is attached to.

Returns:

a newly constructed Condition handle that is attached to the given handle.

6.192.1.2 static void **decaf::internal::util::concurrent::ConditionImpl::destroy**
(**decaf::util::concurrent::ConditionHandle** * *handle*) [static]

Destroy a previously create Condition instance.

Parameters:

handle The Condition handle to be destroyed.

6.192.1.3 static void decaf::internal::util::concurrent::ConditionImpl::notify
(decaf::util::concurrent::ConditionHandle * *condition*) [static]

Signals one Thread that is waiting on this condition to wake up.

Parameters:

condition the handle to the condition to wait on.

6.192.1.4 static void decaf::internal::util::concurrent::ConditionImpl::notifyAll
(decaf::util::concurrent::ConditionHandle * *condition*) [static]

Signals all Threads that is waiting on this condition to wake up.

Parameters:

condition the handle to the condition to wait on.

6.192.1.5 static void decaf::internal::util::concurrent::ConditionImpl::wait
(decaf::util::concurrent::ConditionHandle * *condition*, long long *mills*,
long long *nanos*) [static]

Waits for the condition to be signaled or for the time specified to ellapse.

Parameters:

condition the handle to the condition to wait on.

mills the time in milliseconds to wait for the condition to be signaled.

nanos additional time in nanoseconds to wait for the thread to be signaled.

6.192.1.6 static void decaf::internal::util::concurrent::ConditionImpl::wait
(decaf::util::concurrent::ConditionHandle * *condition*) [static]

Waits for the condition to be signaled.

Parameters:

condition the handle to the condition to wait on.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/ConditionImpl.h

6.193 decaf::net::ConnectException Class Reference

#include <src/main/decaf/net/ConnectException.h> Inheritance diagram for decaf::net::ConnectException:

Public Member Functions

- **ConnectException** () throw ()
Default Constructor.
- **ConnectException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **ConnectException** (const **ConnectException** &ex) throw ()
Copy Constructor.
- **ConnectException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ConnectException** (const std::exception *cause) throw ()
Constructor.
- **ConnectException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ConnectException** * **clone** () const
Clones this exception.
- virtual ~**ConnectException** () throw ()

6.193.1 Constructor & Destructor Documentation

6.193.1.1 decaf::net::ConnectException::ConnectException () throw () [inline]

Default Constructor.

6.193.1.2 decaf::net::ConnectException::ConnectException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.193.1.3 decaf::net::ConnectException::ConnectException (const ConnectException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.193.1.4 decaf::net::ConnectException::ConnectException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.193.1.5 decaf::net::ConnectException::ConnectException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.193.1.6 decaf::net::ConnectException::ConnectException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.193.1.7 `virtual decaf::net::ConnectException::~~ConnectException () throw ()`
[inline, virtual]

6.193.2 Member Function Documentation

6.193.2.1 `virtual ConnectException* decaf::net::ConnectException::clone () const`
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2973).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ConnectException.h`

6.194 cms::Connection Class Reference

The client's connection to its provider.

#include <src/main/cms/Connection.h> Inheritance diagram for cms::Connection:

Public Member Functions

- virtual **~Connection** ()
- virtual void **close** ()=0 throw (CMSEException)
Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).
- virtual const **ConnectionMetaData** * **getMetaData** () const =0 throw (CMSEException)
Gets the metadata for this connection.
- virtual **Session** * **createSession** ()=0 throw (CMSEException)
*Creates an **AUTO_ACKNOWLEDGE Session** (p. 2839).*
- virtual **Session** * **createSession** (**Session::AcknowledgeMode** ackMode)=0 throw (CMSEException)
*Creates a new **Session** (p. 2839) to work for this **Connection** (p. 1131) using the specified acknowledgment mode.*
- virtual std::string **getClientID** () const =0
Get the Client Id for this session.
- virtual **ExceptionListener** * **getExceptionListener** () const =0
Gets the registered Exception Listener for this connection.
- virtual void **setExceptionListener** (**ExceptionListener** *listener)=0
Sets the registered Exception Listener for this connection.

6.194.1 Detailed Description

The client's connection to its provider. Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1237) object.
- It supports an optional **ExceptionListener** (p. 1581) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

Since:

1.0

6.194.2 Constructor & Destructor Documentation

6.194.2.1 `virtual cms::Connection::~~Connection () [inline, virtual]`

6.194.3 Member Function Documentation

6.194.3.1 `virtual void cms::Connection::close () throw (CMSException) [pure virtual]`

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions:

CMSException (p. 1031)

Implements `cms::Closeable` (p. 1021).

Implemented in `activemq::core::ActiveMQConnection` (p. 236).

6.194.3.2 `virtual Session* cms::Connection::createSession (Session::AcknowledgeMode ackMode) throw (CMSException) [pure virtual]`

Creates a new `Session` (p. 2839) to work for this `Connection` (p. 1131) using the specified acknowledgment mode.

Parameters:

ackMode the Acknowledgment Mode to use.

Exceptions:

CMSEException (p. 1031)

Implemented in **activemq::core::ActiveMQConnection** (p. 237).

6.194.3.3 `virtual Session* cms::Connection::createSession () throw (CMSEException) [pure virtual]`

Creates an AUTO_ACKNOWLEDGE Session (p. 2839).

Exceptions:

CMSEException (p. 1031)

Implemented in **activemq::core::ActiveMQConnection** (p. 237).

6.194.3.4 `virtual std::string cms::Connection::getClientID () const [pure virtual]`

Get the Client Id for this session.

Returns:

Client Id String

Implemented in **activemq::core::ActiveMQConnection** (p. 238).

6.194.3.5 `virtual ExceptionListener* cms::Connection::getExceptionListener () const [pure virtual]`

Gets the registered Exception Listener for this connection.

Returns:

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 239).

6.194.3.6 `virtual const ConnectionMetaData* cms::Connection::getMetaData () const throw (CMSEException) [pure virtual]`

Gets the metadata for this connection.

Returns:

the connection MetaData pointer (caller does not own it).

Exceptions:

CMSEException (p. 1031) if the provider fails to get the connection metadata for this connection.

See also:

ConnectionMetaData (p. 1237)

Since:

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 239).

6.194.3.7 **virtual void cms::Connection::setExceptionListener** (**ExceptionListener ***
listener) [pure virtual]

Sets the registered Exception Listener for this connection.

Parameters:

listener pointer to and ExceptionListener (p. 1581)

Implemented in **activemq::core::ActiveMQConnection** (p. 241).

The documentation for this class was generated from the following file:

- `src/main/cms/Connection.h`

6.195 activemq::commands::ConnectionControl Class Reference

#include <src/main/activemq/commands/ConnectionControl.h> Inheritance diagram for activemq::commands::ConnectionControl:

Public Member Functions

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionControl * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual bool **isExit** () const
- virtual void **setExit** (bool exit)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isResume** () const
- virtual void **setResume** (bool resume)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool suspend)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONCONTROL** = 18

Protected Member Functions

- **ConnectionControl** (const **ConnectionControl** &)
- **ConnectionControl** & **operator=** (const **ConnectionControl** &)

Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**

6.195.1 Constructor & Destructor Documentation

6.195.1.1 **activemq::commands::ConnectionControl::ConnectionControl** (const **ConnectionControl** &) [inline, protected]

6.195.1.2 **activemq::commands::ConnectionControl::ConnectionControl** ()

6.195.1.3 **virtual activemq::commands::ConnectionControl::~~ConnectionControl** () [virtual]

6.195.2 Member Function Documentation

6.195.2.1 **virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.195.2.2 **virtual void activemq::commands::ConnectionControl::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.195.2.3 virtual bool activemq::commands::ConnectionControl::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.195.2.4 virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.195.2.5 `virtual bool activemq::commands::ConnectionControl::isClose () const`
[virtual]
- 6.195.2.6 `virtual bool activemq::commands::ConnectionControl::isExit () const`
[virtual]
- 6.195.2.7 `virtual bool activemq::commands::ConnectionControl::isFaultTolerant ()`
`const` [virtual]
- 6.195.2.8 `virtual bool activemq::commands::ConnectionControl::isResume () const`
[virtual]
- 6.195.2.9 `virtual bool activemq::commands::ConnectionControl::isSuspend () const`
[virtual]
- 6.195.2.10 `ConnectionControl& activemq::commands::ConnectionControl::operator= (const`
`ConnectionControl &) [inline, protected]`
- 6.195.2.11 `virtual void activemq::commands::ConnectionControl::setClose (bool`
`close) [virtual]`
- 6.195.2.12 `virtual void activemq::commands::ConnectionControl::setExit (bool`
`exit) [virtual]`
- 6.195.2.13 `virtual void activemq::commands::ConnectionControl::setFaultTolerant`
`(bool faultTolerant) [virtual]`
- 6.195.2.14 `virtual void activemq::commands::ConnectionControl::setResume (bool`
`resume) [virtual]`
- 6.195.2.15 `virtual void activemq::commands::ConnectionControl::setSuspend (bool`
`suspend) [virtual]`
- 6.195.2.16 `virtual std::string activemq::commands::ConnectionControl::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p.655).

- 6.195.2.17 `virtual Pointer<Command> activemq::commands::ConnectionControl::visit`
`(activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.195.3 Field Documentation

6.195.3.1 `bool activemq::commands::ConnectionControl::close` [protected]

6.195.3.2 `bool activemq::commands::ConnectionControl::exit` [protected]

6.195.3.3 `bool activemq::commands::ConnectionControl::faultTolerant` [protected]

6.195.3.4 `const unsigned char activemq::commands::ConnectionControl::ID_ - CONNECTIONCONTROL = 18` [static]

6.195.3.5 `bool activemq::commands::ConnectionControl::resume` [protected]

6.195.3.6 `bool activemq::commands::ConnectionControl::suspend` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionControl.h`

6.196 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1140).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.196.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1140). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.196.2 Constructor & Destructor Documentation

6.196.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

6.196.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

6.196.3 Member Function Documentation

6.196.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.196.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.196.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.196.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.196.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.196.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.196.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h

6.197 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1144).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.197.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1144). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.197.2 Constructor & Destructor Documentation

6.197.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

6.197.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

6.197.3 Member Function Documentation

6.197.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.197.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.197.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.197.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.197.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.197.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.197.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h

6.198 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1148).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.198.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1148). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.198.2 Constructor & Destructor Documentation

6.198.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::ConnectionControlMarshaller()` `[inline]`

6.198.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` `[inline, virtual]`

6.198.3 Member Function Documentation

6.198.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.198.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.198.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.198.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.198.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.198.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.198.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h

6.199 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1152).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.199.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1152). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.199.2 Constructor & Destructor Documentation

6.199.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

6.199.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

6.199.3 Member Function Documentation

6.199.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.199.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.199.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.199.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.199.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.199.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.199.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h

6.200 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1156).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.200.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1156). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.200.2 Constructor & Destructor Documentation

6.200.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

6.200.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

6.200.3 Member Function Documentation

6.200.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.200.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.200.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.200.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.200.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.200.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.200.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h

6.201 activemq::commands::ConnectionError Class Reference

#include <src/main/activemq/commands/ConnectionError.h> Inheritance diagram for activemq::commands::ConnectionError:

Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionError** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONERROR** = 16

Protected Member Functions

- **ConnectionError** (const **ConnectionError** &)
- **ConnectionError** & **operator=** (const **ConnectionError** &)

Protected Attributes

- `Pointer< BrokerError > exception`
- `Pointer< ConnectionId > connectionId`

6.201.1 Constructor & Destructor Documentation

6.201.1.1 `activemq::commands::ConnectionError::ConnectionError (const ConnectionError &) [inline, protected]`

6.201.1.2 `activemq::commands::ConnectionError::ConnectionError ()`

6.201.1.3 `virtual activemq::commands::ConnectionError::~~ConnectionError () [virtual]`

6.201.2 Member Function Documentation

6.201.2.1 `virtual ConnectionError* activemq::commands::ConnectionError::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.201.2.2 `virtual void activemq::commands::ConnectionError::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.201.2.3 `virtual bool activemq::commands::ConnectionError::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

- 6.201.2.4 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ()`
[virtual]
- 6.201.2.5 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ()`
const [virtual]
- 6.201.2.6 `virtual unsigned char activemq::commands::ConnectionError::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.201.2.7 `virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException ()`
[virtual]
- 6.201.2.8 `virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException ()`
const [virtual]
- 6.201.2.9 `ConnectionError& activemq::commands::ConnectionError::operator=`
(const ConnectionError &) [inline, protected]
- 6.201.2.10 `virtual void activemq::commands::ConnectionError::setConnectionId`
(const Pointer< ConnectionId > & *connectionId*) [virtual]
- 6.201.2.11 `virtual void activemq::commands::ConnectionError::setException (const`
Pointer< BrokerError > & *exception*) [virtual]
- 6.201.2.12 `virtual std::string activemq::commands::ConnectionError::toString ()`
const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.201.2.13 `virtual Pointer<Command> activemq::commands::ConnectionError::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.201.3 Field Documentation

6.201.3.1 `Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId` [protected]

6.201.3.2 `Pointer<BrokerError> activemq::commands::ConnectionError::exception` [protected]

6.201.3.3 `const unsigned char activemq::commands::ConnectionError::ID _ - CONNECTIONERROR = 16` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionError.h`

6.202 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1164).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.202.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1164). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.202.2 Constructor & Destructor Documentation

6.202.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

6.202.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

6.202.3 Member Function Documentation

6.202.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.202.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.202.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.202.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.202.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.202.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.202.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

6.203 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1168).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.203.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1168). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.203.2 Constructor & Destructor Documentation

6.203.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

6.203.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

6.203.3 Member Function Documentation

6.203.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.203.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.203.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.203.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.203.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.203.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.203.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h

6.204 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1172).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.204.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1172). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.204.2 Constructor & Destructor Documentation

6.204.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

6.204.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

6.204.3 Member Function Documentation

6.204.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.204.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.204.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.204.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.204.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.204.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.204.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h

6.205 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1176).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.205.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1176). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.205.2 Constructor & Destructor Documentation

6.205.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

6.205.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

6.205.3 Member Function Documentation

6.205.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.205.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.205.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.205.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.205.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.205.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.205.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h

6.206 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1180).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.206.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1180). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.206.2 Constructor & Destructor Documentation

6.206.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErrorMarshaller()` [inline]

6.206.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller()` [inline, virtual]

6.206.3 Member Function Documentation

6.206.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.206.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.206.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.206.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.206.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.206.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.206.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h

6.207 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p.1131) objects returned implement the CMS **Connection** (p.1131) interface and hide the CMS Provider specific implementation details behind that interface.

#include <src/main/cms/ConnectionFactory.h> Inheritance diagram for cms::ConnectionFactory:

Public Member Functions

- virtual **~ConnectionFactory** ()
- virtual **Connection * createConnection** ()=0 throw (CMSEException)
Creates a connection with the default user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password)=0 throw (cms::CMSEException)
Creates a connection with the default specified identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0 throw (cms::CMSEException)
Creates a connection with the specified user identity.

Static Public Member Functions

- static **ConnectionFactory * createCMSConnectionFactory** (const std::string &brokerURI) throw (cms::CMSEException)
Static method that is used to create a provider specific connection factory.

6.207.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p.1131) objects returned implement the CMS **Connection** (p.1131) interface and hide the CMS Provider specific implementation details behind that interface. A Client creates a new **ConnectionFactory** (p.1184) either directly by instantiating the provider specific implementation of the factory or by using the static method **createCMSConnectionFactory** which all providers are required to implement.

Since:

1.0

6.207.2 Constructor & Destructor Documentation

6.207.2.1 `virtual cms::ConnectionFactory::~~ConnectionFactory () [inline, virtual]`

6.207.3 Member Function Documentation

6.207.3.1 `static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory (const std::string & brokerURI) throw (cms::CMSException) [static]`

Static method that is used to create a provider specific connection factory. The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p.1184) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

Parameters:

brokerURI The remote address to use to connect to the Provider.

Returns:

A pointer to a provider specific implementation of the **ConnectionFactory** (p.1184) interface, the caller is responsible for deleting this resource.

Exceptions:

CMSException (p. 1031) if an internal error occurs while creating the **ConnectionFactory** (p.1184).

6.207.3.2 `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) throw (cms::CMSException) [pure virtual]`

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p.3014) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

clientId The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions:

CMSException (p. 1031) if an internal error occurs while creating the **Connection** (p.1131).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 246).

6.207.3.3 `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password) throw (cms::CMSEException)` [pure virtual]

Creates a connection with the default specified identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3014) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions:

CMSEException (p. 1031) if an internal error occurs while creating the **Connection** (p. 1131).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 246).

6.207.3.4 `virtual Connection* cms::ConnectionFactory::createConnection () throw (CMSEException)` [pure virtual]

Creates a connection with the default user identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3014) method is explicitly called.

Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions:

CMSEException (p. 1031) if an internal error occurs while creating the **Connection** (p. 1131).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 246).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

6.208 activemq::commands::ConnectionId Class Reference

#include <src/main/activemq/commands/ConnectionId.h> Inheritance diagram for activemq::commands::ConnectionId:

Public Types

- typedef decaf::lang::PointerComparator< ConnectionId > COMPARATOR

Public Member Functions

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** *sessionId)
- **ConnectionId** (const **ProducerId** *producerId)
- **ConnectionId** (const **ConsumerId** *consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **ConnectionId** &value) const
- virtual bool **equals** (const **ConnectionId** &value) const
- virtual bool **operator==** (const **ConnectionId** &value) const
- virtual bool **operator<** (const **ConnectionId** &value) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

Static Public Attributes

- static const unsigned char **ID_CONNECTIONID** = 120

Protected Attributes

- `std::string` `value`

6.208.1 Member Typedef Documentation

6.208.1.1 `typedef decaf::lang::PointerComparator<ConnectionId>`
`activemq::commands::ConnectionId::COMPARATOR`

6.208.2 Constructor & Destructor Documentation

6.208.2.1 `activemq::commands::ConnectionId::ConnectionId ()`

6.208.2.2 `activemq::commands::ConnectionId::ConnectionId (const ConnectionId`
`& other)`

6.208.2.3 `activemq::commands::ConnectionId::ConnectionId (const SessionId *`
`sessionId)`

6.208.2.4 `activemq::commands::ConnectionId::ConnectionId (const ProducerId *`
`producerId)`

6.208.2.5 `activemq::commands::ConnectionId::ConnectionId (const ConsumerId *`
`consumerId)`

6.208.2.6 `virtual activemq::commands::ConnectionId::~~ConnectionId ()` [virtual]

6.208.3 Member Function Documentation

6.208.3.1 `virtual ConnectionId* ac-`
`tivemq::commands::ConnectionId::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.208.3.2 `virtual int activemq::commands::ConnectionId::compareTo (const`
`ConnectionId & value) const` [virtual]

6.208.3.3 `virtual void activemq::commands::ConnectionId::copyDataStructure`
`(const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.208.3.4 `virtual bool activemq::commands::ConnectionId::equals (const ConnectionId & value) const` [virtual]

6.208.3.5 `virtual bool activemq::commands::ConnectionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.208.3.6 `virtual unsigned char activemq::commands::ConnectionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.208.3.7 `virtual std::string& activemq::commands::ConnectionId::getValue ()` [virtual]

6.208.3.8 `virtual const std::string& activemq::commands::ConnectionId::getValue () const` [virtual]

6.208.3.9 `virtual bool activemq::commands::ConnectionId::operator< (const ConnectionId & value) const` [virtual]

6.208.3.10 `ConnectionId& activemq::commands::ConnectionId::operator= (const ConnectionId & other)`

6.208.3.11 `virtual bool activemq::commands::ConnectionId::operator== (const ConnectionId & value) const` [virtual]

6.208.3.12 `virtual void activemq::commands::ConnectionId::setValue (const std::string & value)` [virtual]

6.208.3.13 `virtual std::string activemq::commands::ConnectionId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 718).

6.208.4 Field Documentation

6.208.4.1 `const unsigned char activemq::commands::ConnectionId::ID_ - CONNECTIONID = 120` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.208.4.2 `std::string activemq::commands::ConnectionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

6.209 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1191).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.209.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1191). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.209.2 Constructor & Destructor Documentation

6.209.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.209.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.209.3 Member Function Documentation

6.209.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.209.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.209.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.209.3.4 virtual void `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.209.3.5 virtual int `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.209.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.209.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h

6.210 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1195).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.210.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1195). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.210.2 Constructor & Destructor Documentation

6.210.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.210.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.210.3 Member Function Documentation

6.210.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.210.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.210.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.210.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.210.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.210.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.210.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

6.211 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1199).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.211.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1199). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.211.2 Constructor & Destructor Documentation

6.211.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.211.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.211.3 Member Function Documentation

6.211.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.211.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.211.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.211.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.211.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.211.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.211.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

6.212 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1203).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.212.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1203). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.212.2 Constructor & Destructor Documentation

6.212.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.212.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.212.3 Member Function Documentation

6.212.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.212.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.212.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.212.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.212.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.212.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.212.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h

6.213 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1207).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.213.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1207). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.213.2 Constructor & Destructor Documentation

6.213.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.213.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.213.3 Member Function Documentation

6.213.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.213.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.213.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.213.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.213.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.213.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.213.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h

6.214 activemq::commands::ConnectionInfo Class Reference

#include <src/main/activemq/commands/ConnectionInfo.h> Inheritance diagram for activemq::commands::ConnectionInfo:

Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool brokerMasterConnector)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool manageable)
- virtual bool **isClientMaster** () const

- virtual void **setClientMaster** (bool **clientMaster**)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _ CONNECTIONINFO** = 3

Protected Member Functions

- **ConnectionInfo** (const **ConnectionInfo** &)
- **ConnectionInfo** & **operator=** (const **ConnectionInfo** &)

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**

6.214.1 Constructor & Destructor Documentation

6.214.1.1 **activemq::commands::ConnectionInfo::ConnectionInfo** (const **ConnectionInfo** &) [inline, protected]

6.214.1.2 **activemq::commands::ConnectionInfo::ConnectionInfo** ()

6.214.1.3 **virtual activemq::commands::ConnectionInfo::~~ConnectionInfo** () [virtual]

6.214.2 Member Function Documentation

6.214.2.1 **virtual ConnectionInfo* activemq::commands::ConnectionInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.214.2.2 virtual void activemq::commands::ConnectionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.214.2.3 virtual bool activemq::commands::ConnectionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.214.2.4 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () [virtual]

6.214.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () const [virtual]

6.214.2.6 virtual std::string& activemq::commands::ConnectionInfo::getClientId () [virtual]

6.214.2.7 virtual const std::string& activemq::commands::ConnectionInfo::getClientId () const [virtual]

6.214.2.8 virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () [virtual]

6.214.2.9 virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () const [virtual]

6.214.2.10 virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.214.2.11 `virtual std::string& activemq::commands::ConnectionInfo::getPassword () [virtual]`

- 6.214.2.12 `virtual const std::string& activemq::commands::ConnectionInfo::getPassword () const [virtual]`

- 6.214.2.13 `virtual std::string& activemq::commands::ConnectionInfo::getUserName () [virtual]`

- 6.214.2.14 `virtual const std::string& activemq::commands::ConnectionInfo::getUserName () const [virtual]`

- 6.214.2.15 `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector () const [virtual]`

- 6.214.2.16 `virtual bool activemq::commands::ConnectionInfo::isClientMaster () const [virtual]`

- 6.214.2.17 `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo () const [inline, virtual]`

Returns:

an answer of true to the `isConnectionInfo()` (p. 1214) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 653).

- 6.214.2.18 `virtual bool activemq::commands::ConnectionInfo::isManageable () const [virtual]`
- 6.214.2.19 `ConnectionInfo& activemq::commands::ConnectionInfo::operator= (const ConnectionInfo &) [inline, protected]`
- 6.214.2.20 `virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector (bool brokerMasterConnector) [virtual]`
- 6.214.2.21 `virtual void activemq::commands::ConnectionInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath) [virtual]`
- 6.214.2.22 `virtual void activemq::commands::ConnectionInfo::setClientId (const std::string & clientId) [virtual]`
- 6.214.2.23 `virtual void activemq::commands::ConnectionInfo::setClientMaster (bool clientMaster) [virtual]`
- 6.214.2.24 `virtual void activemq::commands::ConnectionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.214.2.25 `virtual void activemq::commands::ConnectionInfo::setManageable (bool manageable) [virtual]`
- 6.214.2.26 `virtual void activemq::commands::ConnectionInfo::setPassword (const std::string & password) [virtual]`
- 6.214.2.27 `virtual void activemq::commands::ConnectionInfo::setUserName (const std::string & userName) [virtual]`
- 6.214.2.28 `virtual std::string activemq::commands::ConnectionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

- 6.214.2.29 `virtual Pointer<Command> activemq::commands::ConnectionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.1067).

6.214.3 Field Documentation

- 6.214.3.1 `bool activemq::commands::ConnectionInfo::brokerMasterConnector` [protected]
- 6.214.3.2 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::ConnectionInfo::brokerPath` [protected]
- 6.214.3.3 `std::string activemq::commands::ConnectionInfo::clientId` [protected]
- 6.214.3.4 `bool activemq::commands::ConnectionInfo::clientMaster` [protected]
- 6.214.3.5 `Pointer<ConnectionId>` `activemq::commands::ConnectionInfo::connectionId` [protected]
- 6.214.3.6 `const unsigned char activemq::commands::ConnectionInfo::ID_ - CONNECTIONINFO = 3` [static]
- 6.214.3.7 `bool activemq::commands::ConnectionInfo::manageable` [protected]
- 6.214.3.8 `std::string activemq::commands::ConnectionInfo::password` [protected]
- 6.214.3.9 `std::string activemq::commands::ConnectionInfo::userName` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`

6.215 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1217).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.215.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1217). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.215.2 Constructor & Destructor Documentation

6.215.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.215.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.215.3 Member Function Documentation

6.215.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.215.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.215.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.215.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.215.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.215.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.215.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h

6.216 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1221).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.216.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1221). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.216.2 Constructor & Destructor Documentation

6.216.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.216.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.216.3 Member Function Documentation

6.216.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.216.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.216.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 672).

6.216.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 673).

6.216.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 674).

6.216.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.216.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h

6.217 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1225).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.217.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1225). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.217.2 Constructor & Destructor Documentation

6.217.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoM
()` [inline]

6.217.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~~ConnectionInfo
()` [inline, virtual]

6.217.3 Member Function Documentation

6.217.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject
()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.217.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureT
()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.217.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 665).

6.217.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 666).

6.217.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 667).

6.217.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.217.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h

6.218 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1229).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.218.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1229). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.218.2 Constructor & Destructor Documentation

6.218.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.218.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.218.3 Member Function Documentation

6.218.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.218.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.218.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.218.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.218.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.218.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.218.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h

6.219 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1233).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.219.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1233). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.219.2 Constructor & Destructor Documentation

6.219.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.219.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.219.3 Member Function Documentation

6.219.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.219.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.219.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.219.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.219.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.219.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.219.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h

6.220 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p.1237) object provides information describing the **Connection** (p.1131) object.

#include <src/main/cms/ConnectionMetaData.h> Inheritance diagram for cms::ConnectionMetaData:

Public Member Functions

- virtual **~ConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const =0 throw (cms::CMSEException)
Gets the CMS API version.
- virtual int **getCMSMajorVersion** () const =0 throw (cms::CMSEException)
Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () const =0 throw (cms::CMSEException)
Gets the CMS minor version number.
- virtual std::string **getCMSProviderName** () const =0 throw (cms::CMSEException)
Gets the CMS provider name.
- virtual std::string **getProviderVersion** () const =0 throw (cms::CMSEException)
Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () const =0 throw (cms::CMSEException)
Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () const =0 throw (cms::CMSEException)
Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () const =0 throw (cms::CMSEException)
Gets an Vector of the CMSX property names.

6.220.1 Detailed Description

A **ConnectionMetaData** (p.1237) object provides information describing the **Connection** (p.1131) object.

Since:

1.3

6.220.2 Constructor & Destructor Documentation

6.220.2.1 `virtual cms::ConnectionMetaData::~~ConnectionMetaData () [inline, virtual]`

6.220.3 Member Function Documentation

6.220.3.1 `virtual int cms::ConnectionMetaData::getCMSMajorVersion () const throw (cms::CMSException) [pure virtual]`

Gets the CMS major version number.

Returns:

the CMS API major version number

Exceptions:

CMSException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 250).

6.220.3.2 `virtual int cms::ConnectionMetaData::getCMSMinorVersion () const throw (cms::CMSException) [pure virtual]`

Gets the CMS minor version number.

Returns:

the CMS API minor version number

Exceptions:

CMSException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 250).

6.220.3.3 `virtual std::string cms::ConnectionMetaData::getCMSProviderName () const throw (cms::CMSException) [pure virtual]`

Gets the CMS provider name.

Returns:

the CMS provider name

Exceptions:

CMSException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 250).

6.220.3.4 `virtual std::string cms::ConnectionMetaData::getCMSVersion () const
throw (cms::CMSEException) [pure virtual]`

Gets the CMS API version.

Returns:

the CMS API Version in String form.

Exceptions:

CMSEException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 251).

6.220.3.5 `virtual std::vector<std::string>
cms::ConnectionMetaData::getCMSXPropertyNames ()
const throw (cms::CMSEException) [pure virtual]`

Gets an Vector of the CMSX property names.

Returns:

an Vector of CMSX property names

Exceptions:

CMSEException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 251).

6.220.3.6 `virtual int cms::ConnectionMetaData::getProviderMajorVersion () const
throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider major version number.

Returns:

the CMS provider major version number

Exceptions:

CMSEException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 251).

6.220.3.7 `virtual int cms::ConnectionMetaData::getProviderMinorVersion () const
throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider minor version number.

Returns:

the CMS provider minor version number

Exceptions:

CMSEException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 252).

6.220.3.8 `virtual std::string cms::ConnectionMetaData::getProviderVersion ()
const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider version.

Returns:

the CMS provider version

Exceptions:

CMSEException (p. 1031) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 252).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

6.221 activemq::state::ConnectionState Class Reference

```
#include <src/main/activemq/state/ConnectionState.h>
```

Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual **~ConnectionState** ()
- **std::string toString** () const
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- **std::vector**< **Pointer**< **TransactionState** > > **getTransactionStates** () const
- **Pointer**< **TransactionState** > **removeTransactionState** (const **Pointer**< **TransactionId** > &id)
- void **addSession** (const **Pointer**< **SessionInfo** > &info)
- **Pointer**< **SessionState** > **removeSession** (const **Pointer**< **SessionId** > &id)
- const **Pointer**< **SessionState** > & **getSessionState** (const **Pointer**< **SessionId** > &id) const
- const **StlList**< **Pointer**< **DestinationInfo** > > & **getTempDesinations** () const
- **std::vector**< **Pointer**< **SessionState** > > **getSessionStates** () const

6.221.1 Constructor & Destructor Documentation

- 6.221.1.1 `activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & info)`
- 6.221.1.2 `virtual activemq::state::ConnectionState::~~ConnectionState ()` [virtual]

6.221.2 Member Function Documentation

- 6.221.2.1 `void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & info)` [inline]
- 6.221.2.2 `void activemq::state::ConnectionState::addTempDestination (const Pointer< DestinationInfo > & info)` [inline]
- 6.221.2.3 `void activemq::state::ConnectionState::addTransactionState (const Pointer< TransactionId > & id)` [inline]
- 6.221.2.4 `void activemq::state::ConnectionState::checkShutdown () const`
- 6.221.2.5 `const Pointer< commands::ConnectionInfo > & activemq::state::ConnectionState::getInfo () const` [inline]
- 6.221.2.6 `const Pointer< SessionState > & activemq::state::ConnectionState::getSessionState (const Pointer< SessionId > & id) const` [inline]
- 6.221.2.7 `std::vector< Pointer< SessionState > > activemq::state::ConnectionState::getSessionStates () const` [inline]
- 6.221.2.8 `const StlList< Pointer< DestinationInfo > > & activemq::state::ConnectionState::getTempDesinations () const` [inline]
- 6.221.2.9 `const Pointer< TransactionState > & activemq::state::ConnectionState::getTransactionState (const Pointer< TransactionId > & id) const` [inline]
- 6.221.2.10 `std::vector< Pointer< TransactionState > > activemq::state::ConnectionState::getTransactionStates () const` [inline]
- 6.221.2.11 `Pointer< SessionState > activemq::state::ConnectionState::removeSession (const Pointer< SessionId > & id)` [inline]
- 6.221.2.12 `void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & destination)` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.221.2.13 `Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & id) [inline]`
- 6.221.2.14 `void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & info)`
- 6.221.2.15 `void activemq::state::ConnectionState::shutdown ()`
- 6.221.2.16 `std::string activemq::state::ConnectionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionState.h`

6.222 activemq::state::ConnectionStateTracker Class Reference

#include <src/main/activemq/state/ConnectionStateTracker.h> Inheritance diagram for activemq::state::ConnectionStateTracker:

Public Member Functions

- **ConnectionStateTracker** ()
- virtual **~ConnectionStateTracker** ()
- **Pointer< Tracked > track** (const **Pointer< Command >** &command) throw (decaf::io::IOException)
- void **trackBack** (const **Pointer< Command >** &command)
- void **restore** (const **Pointer< transport::Transport >** &transport) throw (decaf::io::IOException)
- virtual **Pointer< Command > processDestinationInfo** (**DestinationInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveDestination** (**DestinationInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processProducerInfo** (**ProducerInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveProducer** (**ProducerId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processConsumerInfo** (**ConsumerInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveConsumer** (**ConsumerId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processSessionInfo** (**SessionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveSession** (**SessionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processConnectionInfo** (**ConnectionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveConnection** (**ConnectionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processMessage** (**Message** *message) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processMessageAck** (**MessageAck** *ack) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processBeginTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processPrepareTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processCommitTransactionOnePhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processCommitTransactionTwoPhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRollbackTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)

- virtual **Pointer< Command > processEndTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)

Friends

- class **RemoveTransactionAction**

6.222.1 Constructor & Destructor Documentation

6.222.1.1 `activemq::state::ConnectionStateTracker::ConnectionStateTracker ()`

6.222.1.2 `virtual
activemq::state::ConnectionStateTracker::~~ConnectionStateTracker ()
[virtual]`

6.222.2 Member Function Documentation

6.222.2.1 `int activemq::state::ConnectionStateTracker::getMaxCacheSize () const
[inline]`

6.222.2.2 `bool activemq::state::ConnectionStateTracker::isRestoreConsumers ()
const [inline]`

6.222.2.3 `bool activemq::state::ConnectionStateTracker::isRestoreProducers ()
const [inline]`

6.222.2.4 `bool activemq::state::ConnectionStateTracker::isRestoreSessions () const
[inline]`

6.222.2.5 `bool activemq::state::ConnectionStateTracker::isRestoreTransaction ()
const [inline]`

6.222.2.6 `bool activemq::state::ConnectionStateTracker::isTrackMessages () const
[inline]`

6.222.2.7 `bool activemq::state::ConnectionStateTracker::isTrackTransactions ()
const [inline]`

6.222.2.8 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processBeginTransaction
(TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1071).

6.222.2.9 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processCommitTransactionOnePhase
(TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1071).

6.222.2.10 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase
(TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1071).

6.222.2.11 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1072).

6.222.2.12 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1072).

6.222.2.13 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1072).

6.222.2.14 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1072).

6.222.2.15 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message * *message*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1073).

6.222.2.16 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck * *ack*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1073).

6.222.2.17 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1073).

6.222.2.18 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1074).

6.222.2.19 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection (ConnectionId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1074).

6.222.2.20 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1074).

6.222.2.21 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination (DestinationInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1074).

6.222.2.22 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1074).

6.222.2.23 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1075).

6.222.2.24 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction (TransactionInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1075).

6.222.2.25 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p.1075).

6.222.2.26 void activemq::state::ConnectionStateTracker::restore (const Pointer<transport::Transport > & *transport*) throw (decaf::io::IOException)

6.222.2.27 void activemq::state::ConnectionStateTracker::setMaxCacheSize (int *maxCacheSize*) [inline]

6.222.2.28 void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool *restoreConsumers*) [inline]

6.222.2.29 void activemq::state::ConnectionStateTracker::setRestoreProducers (bool *restoreProducers*) [inline]

6.222.2.30 void activemq::state::ConnectionStateTracker::setRestoreSessions (bool *restoreSessions*) [inline]

6.222.2.31 void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool *restoreTransaction*) [inline]

6.222.2.32 void activemq::state::ConnectionStateTracker::setTrackMessages (bool *trackMessages*) [inline]

6.222.2.33 void activemq::state::ConnectionStateTracker::setTrackTransactions (bool *trackTransactions*) [inline]

6.222.2.34 Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer< Command > & *command*) throw (decaf::io::IOException)

6.222.2.35 void activemq::state::ConnectionStateTracker::trackBack (const Pointer< Command > & *command*)

6.222.3 Friends And Related Function Documentation

6.222.3.1 friend class RemoveTransactionAction [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/state/ConnectionStateTracker.h

6.223 activemq::commands::ConsumerControl Class Reference

#include <src/main/activemq/commands/ConsumerControl.h> Inheritance diagram for activemq::commands::ConsumerControl:

Public Member Functions

- **ConsumerControl** ()
- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerControl * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int prefetch)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool flush)
- virtual bool **isStart** () const
- virtual void **setStart** (bool start)
- virtual bool **isStop** () const
- virtual void **setStop** (bool stop)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERCONTROL** = 17

Protected Member Functions

- **ConsumerControl** (const **ConsumerControl** &)
- **ConsumerControl** & **operator=** (const **ConsumerControl** &)

Protected Attributes

- bool **close**
- **Pointer**< **ConsumerId** > **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

6.223.1 Constructor & Destructor Documentation

6.223.1.1 **activemq::commands::ConsumerControl::ConsumerControl** (const **ConsumerControl** &) [inline, protected]

6.223.1.2 **activemq::commands::ConsumerControl::ConsumerControl** ()

6.223.1.3 **virtual activemq::commands::ConsumerControl::~~ConsumerControl** () [virtual]

6.223.2 Member Function Documentation

6.223.2.1 **virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.223.2.2 **virtual void activemq::commands::ConsumerControl::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.223.2.3 `virtual bool activemq::commands::ConsumerControl::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.223.2.4 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ()` [virtual]

6.223.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () const` [virtual]

6.223.2.6 `virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.223.2.7 `virtual int activemq::commands::ConsumerControl::getPrefetch () const`
[virtual]
- 6.223.2.8 `virtual bool activemq::commands::ConsumerControl::isClose () const`
[virtual]
- 6.223.2.9 `virtual bool activemq::commands::ConsumerControl::isFlush () const`
[virtual]
- 6.223.2.10 `virtual bool activemq::commands::ConsumerControl::isStart () const`
[virtual]
- 6.223.2.11 `virtual bool activemq::commands::ConsumerControl::isStop () const`
[virtual]
- 6.223.2.12 `ConsumerControl& activemq::commands::ConsumerControl::operator=`
(const ConsumerControl &) [inline, protected]
- 6.223.2.13 `virtual void activemq::commands::ConsumerControl::setClose (bool`
close) [virtual]
- 6.223.2.14 `virtual void activemq::commands::ConsumerControl::setConsumerId`
(const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.223.2.15 `virtual void activemq::commands::ConsumerControl::setFlush (bool`
flush) [virtual]
- 6.223.2.16 `virtual void activemq::commands::ConsumerControl::setPrefetch (int`
prefetch) [virtual]
- 6.223.2.17 `virtual void activemq::commands::ConsumerControl::setStart (bool`
start) [virtual]
- 6.223.2.18 `virtual void activemq::commands::ConsumerControl::setStop (bool`
stop) [virtual]
- 6.223.2.19 `virtual std::string activemq::commands::ConsumerControl::toString ()`
const [virtual]

Returns a string containing the information for this **DataSet** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

6.223.2.20 `virtual Pointer<Command> activemq::commands::ConsumerControl::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.223.3 Field Documentation

6.223.3.1 `bool activemq::commands::ConsumerControl::close` [protected]

6.223.3.2 `Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId` [protected]

6.223.3.3 `bool activemq::commands::ConsumerControl::flush` [protected]

6.223.3.4 `const unsigned char activemq::commands::ConsumerControl::ID _- CONSUMERCONTROL = 17` [static]

6.223.3.5 `int activemq::commands::ConsumerControl::prefetch` [protected]

6.223.3.6 `bool activemq::commands::ConsumerControl::start` [protected]

6.223.3.7 `bool activemq::commands::ConsumerControl::stop` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerControl.h`

6.224 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1255).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.224.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1255). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.224.2 Constructor & Destructor Documentation

6.224.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

6.224.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

6.224.3 Member Function Documentation

6.224.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.224.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.224.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.224.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.224.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.224.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.224.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h

6.225 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1259).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1259). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.225.2 Constructor & Destructor Documentation

6.225.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

6.225.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

6.225.3 Member Function Documentation

6.225.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.225.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.225.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.225.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.225.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.225.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.225.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h

6.226 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1263).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.226.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1263). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.226.2 Constructor & Destructor Documentation

6.226.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

6.226.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

6.226.3 Member Function Documentation

6.226.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.226.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.226.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.226.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.226.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.226.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.226.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h

6.227 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1267).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.227.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1267). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.227.2 Constructor & Destructor Documentation

6.227.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

6.227.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

6.227.3 Member Function Documentation

6.227.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.227.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.227.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.227.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.227.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.227.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.227.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h

6.228 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1271).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.228.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1271). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.228.2 Constructor & Destructor Documentation

6.228.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerControlMarshaller()` [inline]

6.228.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~~ConsumerControlMarshaller()` [inline, virtual]

6.228.3 Member Function Documentation

6.228.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.228.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.228.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.228.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.228.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.228.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.228.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h

6.229 activemq::commands::ConsumerId Class Reference

#include <src/main/activemq/commands/ConsumerId.h> Inheritance diagram for activemq::commands::ConsumerId:

Public Types

- typedef decaf::lang::PointerComparator< ConsumerId > COMPARATOR

Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **ConsumerId** &value) const
- virtual bool **equals** (const **ConsumerId** &value) const
- virtual bool **operator==** (const **ConsumerId** &value) const
- virtual bool **operator<** (const **ConsumerId** &value) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)

Static Public Attributes

- static const unsigned char **ID_CONSUMERID** = 122

Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

6.229.1 Member Typedef Documentation

6.229.1.1 `typedef decaf::lang::PointerComparator<ConsumerId>
activemq::commands::ConsumerId::COMPARATOR`

6.229.2 Constructor & Destructor Documentation

6.229.2.1 `activemq::commands::ConsumerId::ConsumerId ()`

6.229.2.2 `activemq::commands::ConsumerId::ConsumerId (const ConsumerId &
other)`

6.229.2.3 `activemq::commands::ConsumerId::ConsumerId (const SessionId &
sessionId, long long consumerId) [inline]`

References `activemq::commands::SessionId::getConnectionId()`, and `activemq::commands::SessionId::getValue()`.

6.229.2.4 `virtual activemq::commands::ConsumerId::~~ConsumerId () [virtual]`

6.229.3 Member Function Documentation

6.229.3.1 `virtual ConsumerId* activemq::commands::ConsumerId::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.229.3.2 `virtual int activemq::commands::ConsumerId::compareTo (const
ConsumerId & value) const [virtual]`

6.229.3.3 `virtual void activemq::commands::ConsumerId::copyDataStructure
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.229.3.4 `virtual bool activemq::commands::ConsumerId::equals (const ConsumerId & value) const` [virtual]

6.229.3.5 `virtual bool activemq::commands::ConsumerId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.229.3.6 `virtual std::string& activemq::commands::ConsumerId::getConnectionId ()` [virtual]

6.229.3.7 `virtual const std::string& activemq::commands::ConsumerId::getConnectionId () const` [virtual]

6.229.3.8 `virtual unsigned char activemq::commands::ConsumerId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.229.3.9 `const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId () const`
- 6.229.3.10 `virtual long long activemq::commands::ConsumerId::getSessionId () const [virtual]`
- 6.229.3.11 `virtual long long activemq::commands::ConsumerId::getValue () const [virtual]`
- 6.229.3.12 `virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId & value) const [virtual]`
- 6.229.3.13 `ConsumerId& activemq::commands::ConsumerId::operator= (const ConsumerId & other)`
- 6.229.3.14 `virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & value) const [virtual]`
- 6.229.3.15 `virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.229.3.16 `virtual void activemq::commands::ConsumerId::setSessionId (long long sessionId) [virtual]`
- 6.229.3.17 `virtual void activemq::commands::ConsumerId::setValue (long long value) [virtual]`
- 6.229.3.18 `virtual std::string activemq::commands::ConsumerId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.718).

6.229.4 Field Documentation

- 6.229.4.1 `std::string activemq::commands::ConsumerId::connectionId [protected]`
- 6.229.4.2 `const unsigned char activemq::commands::ConsumerId::ID _- CONSUMERID = 122 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.229.4.3 `long long activemq::commands::ConsumerId::sessionId` [protected]

6.229.4.4 `long long activemq::commands::ConsumerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerId.h`

6.230 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1280).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.230.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1280). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.230.2 Constructor & Destructor Documentation

6.230.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarshaller()` `[inline]`

6.230.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` `[inline, virtual]`

6.230.3 Member Function Documentation

6.230.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.230.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.230.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.230.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.230.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.230.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.230.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h

6.231 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1284).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.231.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1284). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.231.2 Constructor & Destructor Documentation

6.231.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::ConsumerIdMarshaller()` `[inline]`

6.231.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` `[inline, virtual]`

6.231.3 Member Function Documentation

6.231.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.231.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.231.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.231.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.231.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.231.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.231.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h

6.232 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1288).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.232.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1288). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.232.2 Constructor & Destructor Documentation

6.232.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshaller()` `[inline]`

6.232.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` `[inline, virtual]`

6.232.3 Member Function Documentation

6.232.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.232.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.232.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.232.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.232.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.232.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.232.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h

6.233 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1292).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.233.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1292). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.233.2 Constructor & Destructor Documentation

6.233.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

6.233.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

6.233.3 Member Function Documentation

6.233.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.233.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.233.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.233.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.233.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.233.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.233.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h

6.234 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1296).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.234.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1296). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.234.2 Constructor & Destructor Documentation

6.234.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshaller()` [inline]

6.234.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~~ConsumerIdMarshaller()` [inline, virtual]

6.234.3 Member Function Documentation

6.234.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.234.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.234.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.234.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.234.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.234.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.234.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h

6.235 activemq::commands::ConsumerInfo Class Reference

#include <src/main/activemq/commands/ConsumerInfo.h> Inheritance diagram for activemq::commands::ConsumerInfo:

Public Member Functions

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int prefetchSize)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int maximumPendingMessageLimit)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual bool **isNoLocal** () const

- virtual void **setNoLocal** (bool **noLocal**)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool **retroactive**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **brokerPath**)
- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > & **additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **networkConsumerPath**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _CONSUMERINFO** = 5

Protected Member Functions

- **ConsumerInfo** (const **ConsumerInfo** &)
- **ConsumerInfo** & **operator=** (const **ConsumerInfo** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**

- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- **Pointer< BooleanExpression > additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector< decaf::lang::Pointer< ConsumerId > > **networkConsumerPath**

6.235.1 Constructor & Destructor Documentation

- 6.235.1.1** `activemq::commands::ConsumerInfo::ConsumerInfo (const ConsumerInfo &) [inline, protected]`
- 6.235.1.2** `activemq::commands::ConsumerInfo::ConsumerInfo ()`
- 6.235.1.3** `virtual activemq::commands::ConsumerInfo::~~ConsumerInfo () [virtual]`

6.235.2 Member Function Documentation

- 6.235.2.1** `virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

- 6.235.2.2** `virtual void activemq::commands::ConsumerInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.235.2.3 virtual bool activemq::commands::ConsumerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.235.2.4 virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () [virtual]**6.235.2.5 virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () const [virtual]****6.235.2.6 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () [virtual]****6.235.2.7 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () const [virtual]****6.235.2.8 virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () [virtual]****6.235.2.9 virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () const [virtual]****6.235.2.10 virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType () const [virtual]**

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.235.2.11 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ()`
[virtual]
- 6.235.2.12 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () const` [virtual]
- 6.235.2.13 `virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit () const` [virtual]
- 6.235.2.14 `virtual std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ()`
[virtual]
- 6.235.2.15 `virtual const std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath () const` [virtual]
- 6.235.2.16 `virtual int activemq::commands::ConsumerInfo::getPrefetchSize () const`
[virtual]
- 6.235.2.17 `virtual unsigned char activemq::commands::ConsumerInfo::getPriority () const` [virtual]
- 6.235.2.18 `virtual std::string& activemq::commands::ConsumerInfo::getSelector ()`
[virtual]
- 6.235.2.19 `virtual const std::string& activemq::commands::ConsumerInfo::getSelector () const`
[virtual]
- 6.235.2.20 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ()` [virtual]
- 6.235.2.21 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName () const` [virtual]
- 6.235.2.22 `virtual bool activemq::commands::ConsumerInfo::isBrowser () const`
[virtual]
- 6.235.2.23 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo () const` [inline, virtual]

Returns:

an answer of true to the `isConsumerInfo()` (p. 1304) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 653).

- 6.235.2.24 virtual bool activemq::commands::ConsumerInfo::isDispatchAsync () const [virtual]
- 6.235.2.25 virtual bool activemq::commands::ConsumerInfo::isExclusive () const [virtual]
- 6.235.2.26 virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription () const [virtual]
- 6.235.2.27 virtual bool activemq::commands::ConsumerInfo::isNoLocal () const [virtual]
- 6.235.2.28 virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks () const [virtual]
- 6.235.2.29 virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge () const [virtual]
- 6.235.2.30 virtual bool activemq::commands::ConsumerInfo::isRetroactive () const [virtual]
- 6.235.2.31 ConsumerInfo& activemq::commands::ConsumerInfo::operator= (const ConsumerInfo &) [inline, protected]
- 6.235.2.32 virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate (const Pointer< BooleanExpression > & *additionalPredicate*) [virtual]
- 6.235.2.33 virtual void activemq::commands::ConsumerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]
- 6.235.2.34 virtual void activemq::commands::ConsumerInfo::setBrowser (bool *browser*) [virtual]
- 6.235.2.35 virtual void activemq::commands::ConsumerInfo::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.235.2.36 virtual void activemq::commands::ConsumerInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.235.2.37 virtual void activemq::commands::ConsumerInfo::setDispatchAsync (bool *dispatchAsync*) [virtual]
- 6.235.2.38 virtual void activemq::commands::ConsumerInfo::setExclusive (bool *exclusive*) [virtual]
- 6.235.2.39 virtual void activemq::commands::ConsumerInfo::setMaximumPendingMessageLimit (int *maximumPendingMessageLimit*) [virtual]
- 6.235.2.40 virtual void activemq::commands::ConsumerInfo::setNetworkConsumerPath (const std::vector< decaf::lang::Pointer< ConsumerId > > & *networkConsumerPath*) [virtual]

Generated on Fri Apr 30 14:04:20 2010 for activemq-cpp-3.1.2 by Doxygen

- 6.235.2.41 virtual void activemq::commands::ConsumerInfo::setNetworkSubscription (bool *networkSubscription*) [virtual]

- 6.235.2.42 virtual void activemq::commands::ConsumerInfo::setNoLocal (bool *noLocal*) [virtual]

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

6.235.2.51 `virtual Pointer<Command> activemq::commands::ConsumerInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.235.3 Field Documentation

- 6.235.3.1 `Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate` [protected]
- 6.235.3.2 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath` [protected]
- 6.235.3.3 `bool activemq::commands::ConsumerInfo::browser` [protected]
- 6.235.3.4 `Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId` [protected]
- 6.235.3.5 `Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination` [protected]
- 6.235.3.6 `bool activemq::commands::ConsumerInfo::dispatchAsync` [protected]
- 6.235.3.7 `bool activemq::commands::ConsumerInfo::exclusive` [protected]
- 6.235.3.8 `const unsigned char activemq::commands::ConsumerInfo::ID_ - CONSUMERINFO = 5` [static]
- 6.235.3.9 `int activemq::commands::ConsumerInfo::maximumPendingMessageLimit` [protected]
- 6.235.3.10 `std::vector< decaf::lang::Pointer<ConsumerId> > activemq::commands::ConsumerInfo::networkConsumerPath` [protected]
- 6.235.3.11 `bool activemq::commands::ConsumerInfo::networkSubscription` [protected]
- 6.235.3.12 `bool activemq::commands::ConsumerInfo::noLocal` [protected]
- 6.235.3.13 `bool activemq::commands::ConsumerInfo::noRangeAcks` [protected]
- 6.235.3.14 `bool activemq::commands::ConsumerInfo::optimizedAcknowledge` [protected]
- 6.235.3.15 `int activemq::commands::ConsumerInfo::prefetchSize` [protected]
- 6.235.3.16 `unsigned char activemq::commands::ConsumerInfo::priority` [protected]
- 6.235.3.17 `bool activemq::commands::ConsumerInfo::retroactive` [protected]
- 6.235.3.18 `std::string activemq::commands::ConsumerInfo::selector` [protected]
- 6.235.3.19 `std::string activemq::commands::ConsumerInfo::subscriptionName` [protected]

The documentation for this class was generated from the following file:

Generated on Fri Apr 30 14:04:20 2010 for activemq-cpp-3.1.2 by Doxygen

- `src/main/activemq/commands/ConsumerInfo.h`

6.236 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1309).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.236.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1309). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.236.2 Constructor & Destructor Documentation

6.236.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

6.236.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

6.236.3 Member Function Documentation

6.236.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.236.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.236.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.236.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.236.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.236.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.236.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h

6.237 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1313).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.237.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1313). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.237.2 Constructor & Destructor Documentation

6.237.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

6.237.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

6.237.3 Member Function Documentation

6.237.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.237.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.237.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.237.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.237.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.237.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.237.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h

6.238 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1317).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.238.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1317). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.238.2 Constructor & Destructor Documentation

6.238.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

6.238.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

6.238.3 Member Function Documentation

6.238.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.238.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.238.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.238.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.238.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.238.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.238.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

6.239 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1321).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.239.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1321). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.239.2 Constructor & Destructor Documentation

6.239.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

6.239.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

6.239.3 Member Function Documentation

6.239.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.239.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.239.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.239.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.239.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.239.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.239.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h

6.240 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1325).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.240.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1325). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.240.2 Constructor & Destructor Documentation

6.240.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

6.240.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

6.240.3 Member Function Documentation

6.240.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.240.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.240.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.240.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.240.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.240.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.240.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

6.241 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual **~ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

6.241.1 Constructor & Destructor Documentation

6.241.1.1 **activemq::state::ConsumerState::ConsumerState** (const **Pointer**<
 ConsumerInfo > & *info*)

6.241.1.2 virtual **activemq::state::ConsumerState::~~ConsumerState** () [virtual]

6.241.2 Member Function Documentation

6.241.2.1 const **Pointer**<**ConsumerInfo**>& **activemq::state::ConsumerState::getInfo**
 () const [inline]

6.241.2.2 std::string **activemq::state::ConsumerState::toString** () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConsumerState.h**

6.242 activemq::commands::ControlCommand Class Reference

#include <src/main/activemq/commands/ControlCommand.h> Inheritance diagram for activemq::commands::ControlCommand:

Public Member Functions

- **ControlCommand** ()
- virtual **~ControlCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ControlCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getCommand** () const
- virtual std::string & **getCommand** ()
- virtual void **setCommand** (const std::string &command)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONTROLCOMMAND** = 14

Protected Member Functions

- **ControlCommand** (const **ControlCommand** &)
- **ControlCommand & operator=** (const **ControlCommand** &)

Protected Attributes

- std::string **command**

6.242.1 Constructor & Destructor Documentation

6.242.1.1 `activemq::commands::ControlCommand::ControlCommand (const ControlCommand &) [inline, protected]`

6.242.1.2 `activemq::commands::ControlCommand::ControlCommand ()`

6.242.1.3 `virtual activemq::commands::ControlCommand::~~ControlCommand () [virtual]`

6.242.2 Member Function Documentation

6.242.2.1 `virtual ControlCommand* activemq::commands::ControlCommand::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.242.2.2 `virtual void activemq::commands::ControlCommand::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.242.2.3 `virtual bool activemq::commands::ControlCommand::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

- 6.242.2.4** `virtual std::string& activemq::commands::ControlCommand::getCommand ()`
[virtual]
- 6.242.2.5** `virtual const std::string& activemq::commands::ControlCommand::getCommand ()`
`const` [virtual]
- 6.242.2.6** `virtual unsigned char activemq::commands::ControlCommand::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.242.2.7** `ControlCommand& activemq::commands::ControlCommand::operator=`
`(const ControlCommand &)` [inline, protected]
- 6.242.2.8** `virtual void activemq::commands::ControlCommand::setCommand (const`
`std::string & command)` [virtual]
- 6.242.2.9** `virtual std::string activemq::commands::ControlCommand::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

- 6.242.2.10** `virtual Pointer<Command> activemq::commands::ControlCommand::visit`
`(activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.242.3 Field Documentation

6.242.3.1 `std::string activemq::commands::ControlCommand::command`
[protected]

6.242.3.2 `const unsigned char activemq::commands::ControlCommand::ID_ -
CONTROLCOMMAND = 14` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ControlCommand.h`

6.243 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1334).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.243.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1334). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.243.2 Constructor & Destructor Documentation

6.243.2.1 `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlComm`
`() [inline]`

6.243.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~~ControlComm`
`() [inline, virtual]`

6.243.3 Member Function Documentation

6.243.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.243.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructur`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.243.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.243.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.243.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.243.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.243.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h

6.244 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1338).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.244.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1338). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.244.2 Constructor & Destructor Documentation

6.244.2.1 **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::ControlComm**
() [inline]

6.244.2.2 **virtual**
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]

6.244.3 Member Function Documentation

6.244.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1419).

6.244.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::getDataStructur
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1424).

6.244.3.3 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseMarshal
(**OpenWireFormat * *wireFormat***, **commands::DataStructure * *dataStructure***, **decaf::io::DataOutputStream * *dataOut***) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.244.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.244.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.244.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.244.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h

6.245 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1342).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.245.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1342). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.245.2 Constructor & Destructor Documentation

6.245.2.1 `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlComm
 () [inline]`

6.245.2.2 `virtual
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~~ControlComm
 () [inline, virtual]`

6.245.3 Member Function Documentation

6.245.3.1 `virtual commands::DataStructure* ac-
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject
 () const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.245.3.2 `virtual unsigned char ac-
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructur
 () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.245.3.3 `virtual void ac-
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal
 (OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
 decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.245.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.245.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.245.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.245.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h

6.246 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1346).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.246.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1346). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.246.2 Constructor & Destructor Documentation

6.246.2.1 **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::ControlComm**
() [inline]

6.246.2.2 **virtual**
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]

6.246.3 Member Function Documentation

6.246.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1419).

6.246.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::getDataStructur
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1424).

6.246.3.3 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseMarshal
(**OpenWireFormat * *wireFormat***, **commands::DataStructure * *dataStructure***, **decaf::io::DataOutputStream * *dataOut***) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.246.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.246.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.246.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.246.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h

6.247 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1350).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.247.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1350). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.247.2 Constructor & Destructor Documentation

6.247.2.1 `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlComm
() [inline]`

6.247.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]`

6.247.3 Member Function Documentation

6.247.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.247.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructur
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.247.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.247.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.247.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.247.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.247.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h

6.248 decaf::util::concurrent::CountDownLatch Class Reference

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

Public Member Functions

- **CountDownLatch** (int count)
Constructor.
- virtual **~CountDownLatch** ()
- virtual void **await** () throw (lang::Exception)
Waits for the Count to be zero, and then returns.
- virtual bool **await** (unsigned long timeOut) throw (lang::Exception)
Waits for the Count to hit zero, or a timeout.
- virtual void **countDown** ()
Counts down the latch, releasing all waiting threads when the count hits zero.
- virtual int **getCount** () const
Gets the current count.

6.248.1 Constructor & Destructor Documentation

6.248.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch (int count)

Constructor.

Parameters:

count - number to count down from.

6.248.1.2 virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch () [virtual]

6.248.2 Member Function Documentation

6.248.2.1 virtual bool decaf::util::concurrent::CountDownLatch::await (unsigned long timeOut) throw (lang::Exception) [virtual]

Waits for the Count to hit zero, or a timeout.

Parameters:

timeOut - time in milliseconds to wait.

Returns:

true if the wait made it to count zero, otherwise false

6.248.2.2 virtual void decaf::util::concurrent::CountDownLatch::await () throw (lang::Exception) [virtual]

Waits for the Count to be zero, and then returns.

Exceptions:

Exception

6.248.2.3 virtual void decaf::util::concurrent::CountDownLatch::countDown () [virtual]

Counts down the latch, releasing all waiting threads when the count hits zero.

6.248.2.4 virtual int decaf::util::concurrent::CountDownLatch::getCount () const [inline, virtual]

Gets the current count.

Returns:

int count value

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CountDownLatch.h**

6.249 activemq::commands::DataArrayResponse Class Reference

#include <src/main/activemq/commands/DataArrayResponse.h> Inheritance diagram for activemq::commands::DataArrayResponse:

Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataArrayResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &data)

Static Public Attributes

- static const unsigned char **ID_DATAARRAYRESPONSE** = 33

Protected Member Functions

- **DataArrayResponse** (const **DataArrayResponse** &)
- **DataArrayResponse** & **operator=** (const **DataArrayResponse** &)

Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataStructure** > > **data**

6.249.1 Constructor & Destructor Documentation

6.249.1.1 `activemq::commands::DataArrayResponse::DataArrayResponse (const DataArrayResponse &) [inline, protected]`

6.249.1.2 `activemq::commands::DataArrayResponse::DataArrayResponse ()`

6.249.1.3 `virtual activemq::commands::DataArrayResponse::~~DataArrayResponse () [virtual]`

6.249.2 Member Function Documentation

6.249.2.1 `virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 2782).

6.249.2.2 `virtual void activemq::commands::DataArrayResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::Response` (p. 2782).

6.249.2.3 `virtual bool activemq::commands::DataArrayResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p. 2782).

- 6.249.2.4 `virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () [virtual]`
- 6.249.2.5 `virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () const [virtual]`
- 6.249.2.6 `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from **activemq::commands::Response** (p. 2783).

- 6.249.2.7 `DataArrayResponse& activemq::commands::DataArrayResponse::operator= (const DataArrayResponse &) [inline, protected]`
- 6.249.2.8 `virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & data) [virtual]`
- 6.249.2.9 `virtual std::string activemq::commands::DataArrayResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2783).

6.249.3 Field Documentation

- 6.249.3.1 `std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]`
- 6.249.3.2 `const unsigned char activemq::commands::DataArrayResponse::ID_ - DATAARRAYRESPONSE = 33 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataArrayResponse.h`

6.250 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1359).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.250.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1359). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.250.2 Constructor & Destructor Documentation

6.250.2.1 `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

6.250.2.2 `virtual activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

6.250.3 Member Function Documentation

6.250.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.250.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.250.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.250.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.250.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.250.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2799).

6.250.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2800).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**

6.251 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p.1363).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.251.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p.1363). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.251.2 Constructor & Destructor Documentation

6.251.2.1 `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

6.251.2.2 `virtual activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

6.251.3 Member Function Documentation

6.251.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.251.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.251.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.251.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.251.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.251.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2814).

6.251.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2815).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataArrayResponseMarshaller.h**

6.252 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1367).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.252.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1367). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.252.2 Constructor & Destructor Documentation

6.252.2.1 `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

6.252.2.2 `virtual activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

6.252.3 Member Function Documentation

6.252.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.252.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.252.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.252.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2808).

6.252.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2808).

6.252.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2809).

6.252.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2810).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataArrayResponseMarshaller.h**

6.253 activemq::wireformat::openwire::marshal::v5::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1371).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.253.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1371). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.253.2 Constructor & Destructor Documentation

6.253.2.1 `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

6.253.2.2 `virtual activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

6.253.3 Member Function Documentation

6.253.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.253.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.253.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2802).

6.253.3.4 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseUnmarsh
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (**
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2803).

6.253.3.5 **virtual int ac-**
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure *
***dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException**
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2803).

6.253.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2804).

6.253.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataArrayResponseMarshaller.h**

6.254 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1375).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.254.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1375). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.254.2 Constructor & Destructor Documentation

6.254.2.1 `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayResponseMarshaller()` [inline]

6.254.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~DataArrayResponseMarshaller()` [inline, virtual]

6.254.3 Member Function Documentation

6.254.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.254.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.254.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.254.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.254.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.254.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2794).

6.254.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2795).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataArrayResponseMarshaller.h**

6.255 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

#include <src/main/decaf/io/DataInputStream.h> Inheritance diagram for decaf::io::DataInputStream:

Public Member Functions

- **DataInputStream** (**InputStream** *inputStream, bool own=false)
*Creates a **DataInputStream** (p. 1379) that uses the specified underlying **InputStream** (p. 1740).*
- virtual ~**DataInputStream** ()
- virtual int **read** () throw (**IOException**)
Reads the next byte of data from this input stream.
- virtual int **read** (std::vector< unsigned char > &buffer) throw (**io::IOException**)
Reads some number of bytes from the contained input stream and stores them into the buffer array b.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (**io::IOException**, **lang::exceptions::NullPointerException**)
Reads up to len bytes of data from the contained input stream into an array of bytes.
- virtual bool **readBoolean** () throw (**io::IOException**, **io::EOFException**)
Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** () throw (**io::IOException**, **io::EOFException**)
Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** () throw (**io::IOException**, **io::EOFException**)
Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.
- virtual char **readChar** () throw (**io::IOException**, **io::EOFException**)
Reads an input char and returns the char value.
- virtual double **readDouble** () throw (**io::IOException**, **io::EOFException**)
Reads eight input bytes and returns a double value.
- virtual float **readFloat** () throw (**io::IOException**, **io::EOFException**)
Reads four input bytes and returns a float value.
- virtual int **readInt** () throw (**io::IOException**, **io::EOFException**)
Reads four input bytes and returns an int value.

- virtual long long **readLong** () throw (io::IOException, io::EOFException)
Reads eight input bytes and returns a long value.
- virtual short **readShort** () throw (io::IOException, io::EOFException)
Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** () throw (io::IOException, io::EOFException)
Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** () throw (io::IOException, io::EOFException)
Reads an null terminated ASCII string to the stream and returns the string to the caller.
- virtual std::string **readUTF** () throw (io::IOException, io::EOFException, io::UTFDataFormatException)
Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.
- virtual void **readFully** (std::vector< unsigned char > &buffer) throw (io::IOException, io::EOFException)
Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (io::IOException, io::EOFException, lang::exceptions::NullPointerException)
Reads length bytes from an input stream.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.255.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way. An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1740) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataInputStream (p. 1379) os = new DataInputStream (p. 1379)( new InputStream(), true )
```

Since:

1.0

6.255.2 Constructor & Destructor Documentation

6.255.2.1 decaf::io::DataInputStream::DataInputStream (InputStream * *inputStream*, bool *own* = false)

Creates a **DataInputStream** (p. 1379) that uses the specified underlying **InputStream** (p. 1740).

Parameters:

inputStream the **InputStream** (p. 1740) instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.255.2.2 virtual decaf::io::DataInputStream::~~DataInputStream () [virtual]

6.255.3 Member Function Documentation

6.255.3.1 virtual int decaf::io::DataInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (io::IOException, lang::exceptions::NullPointerException) [virtual]

Reads up to len bytes of data from the contained input stream into an array of bytes. An attempt is made to read as many as len bytes, but a smaller number may be read, possibly zero. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If buffer is null, a NullPointerException is thrown.

If off is negative, or len is negative then an IndexOutOfBoundsException is thrown, if off + len is greater than the allocated length of the array, an **IOException** (p. 1820) will result depending on the platform and compiler settings.

If len is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into buffer.

The first byte read is stored into element b[off], the next one into buffer[off+1], and so on. The number of bytes read is, at most, equal to len. Let k be the number of bytes actually read; these bytes will be stored in elements buffer[off] through buffer[off+k-1], leaving elements buffer[off+k] through buffer[off+len-1] unaffected.

In every case, elements buffer[0] through buffer[off] and elements buffer[off+len] through buffer[buffer.length-1] are unaffected.

If the first byte cannot be read for any reason other than end of file, then an **IOException** (p. 1820) is thrown. In particular, an **IOException** (p. 1820) is thrown if the input stream has been closed.

Parameters:

buffer - byte array to insert read data into

offset - location in buffer to start writing

length - number of bytes to read

Returns:

the total number of bytes read, or -1 if there is no more data because the stream is EOF.

Exceptions:***IOException*** (p. 1820)***NullPointerException*** if the buffer is nullReimplemented from **decaf::io::FilterInputStream** (p.1635).

6.255.3.2 `virtual int decaf::io::DataInputStream::read (std::vector< unsigned char
> & buffer) throw (io::IOException)` [virtual]

Reads some number of bytes from the contained input stream and stores them into the buffer array b. The number of bytes actually read is returned as an integer. This method blocks until input data is available, end of file is detected, or an exception is thrown.

If the length of buffer is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into buffer.

The first byte read is stored into element buffer[0], the next one into buffer[1], and so on. The number of bytes read is, at most, equal to the length of buffer. Let k be the number of bytes actually read; these bytes will be stored in elements b[0] through b[k-1], leaving elements buffer[k] through buffer[buffer.length-1] unaffected.

If the first byte cannot be read for any reason other than end of file, then an **IOException** (p.1820) is thrown. In particular, an **IOException** (p.1820) is thrown if the input stream has been closed.

The read(buffer) method has the same effect as: read(buffer, 0, b.length)

Parameters:

buffer - byte array to insert read data into

Returns:

the total number of bytes read, or -1 if there is no more data because the stream is EOF.

Exceptions:***IOException*** (p. 1820)

6.255.3.3 `virtual int decaf::io::DataInputStream::read () throw (IOException)`
[inline, virtual]

Reads the next byte of data from this input stream. The value byte is returned as an unsigned char in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown. This method simply performs in.read() and returns the result.

Returns:

The next byte.

Exceptions:***IOException*** (p. 1820) thrown if an error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1636).

References **decaf::io::FilterInputStream::read()**.

6.255.3.4 virtual bool decaf::io::DataInputStream::readBoolean () throw (io::IOException, io::EOFException) [virtual]

Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.

Returns:

the boolean value read.

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.5 virtual char decaf::io::DataInputStream::readByte () throw (io::IOException, io::EOFException) [virtual]

Reads and returns one input byte. The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns:

the 8-bit value read.

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.6 virtual char decaf::io::DataInputStream::readChar () throw (io::IOException, io::EOFException) [virtual]

Reads an input char and returns the char value. A ascii char is made up of one bytes. This returns the same result as **readByte**

Returns:

the 8 bit char read

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.7 virtual double decaf::io::DataInputStream::readDouble () throw (io::IOException, io::EOFException) [virtual]

Reads eight input bytes and returns a double value. It does this by first constructing a long long value in exactly the manner of the readlong method, then converting this long value to a double in exactly the manner of the method Double.longBitsToDouble.

Returns:

the double value read

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.8 virtual float decaf::io::DataInputStream::readFloat () throw (io::IOException, io::EOFException) [virtual]

Reads four input bytes and returns a float value. It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float.intBitsToFloat.

Returns:

the float value read

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.9 virtual void decaf::io::DataInputStream::readFully (unsigned char * buffer, std::size_t offset, std::size_t length) throw (io::IOException, io::EOFException, lang::exceptions::NullPointerException) [virtual]

Reads length bytes from an input stream. This method blocks until one of the following conditions occurs: * length bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1571) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1820) other than **EOFException** (p. 1571) is thrown.

If buffer is null, a NullPointerException is thrown. If offset is negative, or len is negative, or offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If len is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to len.

Parameters:

buffer - byte array to insert read data into

offset - location in buffer to start writing

length - number of bytes to read

Exceptions:***IOException*** (p. 1820)***EOFException*** (p. 1571)***NullPointerException*** if the buffer is null

6.255.3.10 `virtual void decaf::io::DataInputStream::readFully (std::vector< unsigned char > & buffer) throw (io::IOException, io::EOFException)` [virtual]

Reads some bytes from an input stream and stores them into the buffer array *buffer*. The number of bytes read is equal to the length of *buffer*.

This method blocks until one of the following conditions occurs: * *buffer.size()* bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1571) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1820) other than **EOFException** (p. 1571) is thrown.

If *buffer.size()* is zero, then no bytes are read. Otherwise, the first byte read is stored into element *b[0]*, the next one into *buffer[1]*, and so on. If an exception is thrown from this method, then it may be that some but not all bytes of *buffer* have been updated with data from the input stream.

Parameters:***buffer*** - vector of char that is read to its size()**Exceptions:*****IOException*** (p. 1820)***EOFException*** (p. 1571)

6.255.3.11 `virtual int decaf::io::DataInputStream::readInt () throw (io::IOException, io::EOFException)` [virtual]

Reads four input bytes and returns an int value. Let *a* be the first byte read, *b* be the second byte, *c* be the third byte, and *d* be the fourth byte. The value returned is:

$$(((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff))$$
Returns:

the int value read

Exceptions:***IOException*** (p. 1820)***EOFException*** (p. 1571)

6.255.3.12 `virtual long long decaf::io::DataInputStream::readLong () throw (io::IOException, io::EOFException)` [virtual]

Reads eight input bytes and returns a long value. Let *a* be the first byte read, *b* be the second byte, *c* be the third byte, *d* be the fourth byte, *e* be the fifth byte, *f* be the sixth byte, *g* be

the seventh byte, and h be the eighth byte. The value returned is: $((\text{long})(a \& 0\text{xff}) \ll 56) | ((\text{long})(b \& 0\text{xff}) \ll 48) | ((\text{long})(c \& 0\text{xff}) \ll 40) | ((\text{long})(d \& 0\text{xff}) \ll 32) | ((\text{long})(e \& 0\text{xff}) \ll 24) | ((\text{long})(f \& 0\text{xff}) \ll 16) | ((\text{long})(g \& 0\text{xff}) \ll 8) | ((\text{long})(h \& 0\text{xff}))$

Returns:

the 64 bit long long read

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.13 virtual short decaf::io::DataInputStream::readShort () throw (io::IOException, io::EOFException) [virtual]

Reads two input bytes and returns a short value. Let a be the first byte read and b be the second byte. The value returned is: $(\text{short})((a \ll 8) | (b \& 0\text{xff}))$

Returns:

the 16 bit short value read

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.14 virtual std::string decaf::io::DataInputStream::readString () throw (io::IOException, io::EOFException) [virtual]

Reads an null terminated ASCII string to the stream and returns the string to the caller.

Returns:

string object containing the string read.

Exceptions:

IOException (p. 1820)

EOFException (p. 1571)

6.255.3.15 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte () throw (io::IOException, io::EOFException) [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns:

the 8 bit unsigned value read

Exceptions:*IOException* (p. 1820)*EOFException* (p. 1571)**6.255.3.16 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort () throw (io::IOException, io::EOFException) [virtual]**

Reads two input bytes and returns an int value in the range 0 through 65535. Let a be the first byte read and b be the second byte. The value returned is: $((a \& 0xff) << 8) | (b \& 0xff)$

Returns:

the 16 bit unsigned short read

Exceptions:*IOException* (p. 1820)*EOFException* (p. 1571)**6.255.3.17 virtual std::string decaf::io::DataInputStream::readUTF () throw (io::IOException, io::EOFException, io::UTFDataFormatException) [virtual]**

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException. This method reads String value written from a Java **DataOutputStream** (p. 1388) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns:

The decoded string read from stream.

Exceptions:*IOException* (p. 1820)*EOFException* (p. 1571)*UTFDataFormatException* (p. 3353)**6.255.3.18 virtual std::size_t decaf::io::DataInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]**

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes. However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1571). The actual number of bytes skipped is returned.

Parameters:

num - number of bytes to skip

Returns:

the total number of bytes skipped

Reimplemented from **decaf::io::FilterInputStream** (p. 1637).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInputStream.h`

6.256 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

#include <src/main/decaf/io/DataOutputStream.h> Inheritance diagram for decaf::io::DataOutputStream:

Public Member Functions

- **DataOutputStream** (**OutputStream** ***outputStream**, bool **own**=false)
Creates a new data output stream to write data to the specified underlying output stream.
- virtual ~**DataOutputStream** ()
- virtual std::size_t **size** () const
Returns the current value of the counter written, the number of bytes written to this data output stream so far.
- virtual void **write** (unsigned char **c**) throw (**IOException**)
Writes a single byte to the output stream.
- virtual void **write** (const unsigned char ***buffer**, std::size_t **offset**, std::size_t **len**) throw (**IOException**, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **write** (const std::vector< unsigned char > &**buffer**) throw (**IOException**)
Writes an array of bytes to the output stream.
- virtual void **writeBoolean** (bool **value**) throw (**IOException**)
Writes a boolean to the underlying output stream as a 1-byte value.
- virtual void **writeByte** (unsigned char **value**) throw (**IOException**)
Writes out a byte to the underlying output stream as a 1-byte value.
- virtual void **writeShort** (short **value**) throw (**IOException**)
Writes a short to the underlying output stream as two bytes, high byte first.
- virtual void **writeUnsignedShort** (unsigned short **value**) throw (**IOException**)
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual void **writeChar** (char **value**) throw (**IOException**)
Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.
- virtual void **writeInt** (int **value**) throw (**IOException**)
Writes an int to the underlying output stream as four bytes, high byte first.
- virtual void **writeLong** (long long **value**) throw (**IOException**)
Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

- virtual void **writeFloat** (float value) throw (IOException)

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

- virtual void **writeDouble** (double value) throw (IOException)

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

- virtual void **writeBytes** (const std::string &value) throw (IOException)

Writes out the string to the underlying output stream as a sequence of bytes.

- virtual void **writeChars** (const std::string &value) throw (IOException)

Writes a string to the underlying output stream as a sequence of characters.

- virtual void **writeUTF** (const std::string &value) throw (IOException, UTFDataFormatException)

Writes out the string to the underlying output stream as a modified UTF-8 encoded sequence of bytes.

Protected Attributes

- std::size_t **written**
- unsigned char **buffer** [8]

6.256.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

6.256.2 Constructor & Destructor Documentation

6.256.2.1 decaf::io::DataOutputStream::DataOutputStream (OutputStream * *outputStream*, bool *own* = false)

Creates a new data output stream to write data to the specified underlying output stream.

Parameters:

outputStream a stream to wrap with this one.

own true if this objects owns the stream that it wraps.

6.256.2.2 virtual decaf::io::DataOutputStream::~~DataOutputStream () [virtual]

6.256.3 Member Function Documentation

6.256.3.1 virtual std::size_t decaf::io::DataOutputStream::size () const [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far. If the counter overflows, it will be wrapped to Integer.MAX_VALUE.

Returns:

the value of the written field.

6.256.3.2 virtual void decaf::io::DataOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from decaf::io::FilterOutputStream (p. 1646).

6.256.3.3 virtual void decaf::io::DataOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream. the counter written is incremented by len.

Parameters:

buffer The array of bytes to write.

offset the position in buffer to start writing from.

len The number of bytes from the buffer to be written.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException if buffer is Null

Reimplemented from decaf::io::FilterOutputStream (p. 1645).

6.256.3.4 virtual void decaf::io::DataOutputStream::write (unsigned char *c*) throw (IOException) [virtual]

Writes a single byte to the output stream. If no exception is thrown, the counter written is incremented by 1.

Parameters:

c the byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1646).

**6.256.3.5 virtual void decaf::io::DataOutputStream::writeBoolean (bool *value*)
throw (IOException) [virtual]**

Writes a boolean to the underlying output stream as a 1-byte value. The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

Parameters:

value the boolean to write.

Exceptions:

IOException (p. 1820)

**6.256.3.6 virtual void decaf::io::DataOutputStream::writeByte (unsigned char
value) throw (IOException) [virtual]**

Writes out a byte to the underlying output stream as a 1-byte value. If no exception is thrown, the counter written is incremented by 1.

Parameters:

value the unsigned char value to write.

Exceptions:

IOException (p. 1820)

**6.256.3.7 virtual void decaf::io::DataOutputStream::writeBytes (const std::string &
value) throw (IOException) [virtual]**

Writes out the string to the underlying output stream as a sequence of bytes. Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.8 virtual void decaf::io::DataOutputStream::writeChar (char *value*) throw (IOException) [virtual]

Writes out a char to the underlying output stream as a one byte value. If no exception is thrown, the counter written is incremented by 1.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.9 virtual void decaf::io::DataOutputStream::writeChars (const std::string & *value*) throw (IOException) [virtual]

Writes a string to the underlying output stream as a sequence of characters. Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.10 virtual void decaf::io::DataOutputStream::writeDouble (double *value*) throw (IOException) [virtual]

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first. If no exception is thrown, the counter written is incremented by 8.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.11 virtual void decaf::io::DataOutputStream::writeFloat (float *value*) throw (IOException) [virtual]

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first. If no exception is thrown, the counter written is incremented by 4.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.12 `virtual void decaf::io::DataOutputStream::writeInt (int value) throw (IOException) [virtual]`

Writes an int to the underlying output stream as four bytes, high byte first. If no exception is thrown, the counter written is incremented by 4.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.13 `virtual void decaf::io::DataOutputStream::writeLong (long long value) throw (IOException) [virtual]`

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first. If no exception is thrown, the counter written is incremented by 8.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.14 `virtual void decaf::io::DataOutputStream::writeShort (short value) throw (IOException) [virtual]`

Writes a short to the underlying output stream as two bytes, high byte first. If no exception is thrown, the counter written is incremented by 2.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820)

6.256.3.15 virtual void decaf::io::DataOutputStream::writeUnsignedShort
(unsigned short *value*) throw (IOException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value - unsigned short to write to the stream

Exceptions:

IOException (p. 1820)

6.256.3.16 virtual void decaf::io::DataOutputStream::writeUTF (const std::string
& *value*) throw (IOException, UTFDataFormatException) [virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes. The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

Parameters:

value the value to write.

Exceptions:

IOException (p. 1820) - on a write error

UTFDataFormatException (p. 3353) - if encoded size if greater than 65535

6.256.4 Field Documentation**6.256.4.1** unsigned char decaf::io::DataOutputStream::buffer[8] [protected]**6.256.4.2** std::size_t decaf::io::DataOutputStream::written [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutputStream.h**

6.257 activemq::commands::DataResponse Class Reference

#include <src/main/activemq/commands/DataResponse.h> Inheritance diagram for activemq::commands::DataResponse:

Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getData** () const
- virtual **Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

Static Public Attributes

- static const unsigned char **ID_DATARESPONSE** = 32

Protected Member Functions

- **DataResponse** (const **DataResponse** &)
- **DataResponse** & **operator=** (const **DataResponse** &)

Protected Attributes

- **Pointer**< **DataStructure** > **data**

6.257.1 Constructor & Destructor Documentation

6.257.1.1 `activemq::commands::DataResponse::DataResponse (const DataResponse &) [inline, protected]`

6.257.1.2 `activemq::commands::DataResponse::DataResponse ()`

6.257.1.3 `virtual activemq::commands::DataResponse::~~DataResponse () [virtual]`

6.257.2 Member Function Documentation

6.257.2.1 `virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 2782).

6.257.2.2 `virtual void activemq::commands::DataResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::Response` (p. 2782).

6.257.2.3 `virtual bool activemq::commands::DataResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p. 2782).

- 6.257.2.4 `virtual Pointer<DataStructure>& activemq::commands::DataResponse::getData ()`
[virtual]
- 6.257.2.5 `virtual const Pointer<DataStructure>& activemq::commands::DataResponse::getData () const`
[virtual]
- 6.257.2.6 `virtual unsigned char activemq::commands::DataResponse::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from **activemq::commands::Response** (p. 2783).

- 6.257.2.7 `DataResponse& activemq::commands::DataResponse::operator= (const DataResponse &)` [inline, protected]
- 6.257.2.8 `virtual void activemq::commands::DataResponse::setData (const Pointer< DataStructure > & data)` [virtual]
- 6.257.2.9 `virtual std::string activemq::commands::DataResponse::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2783).

6.257.3 Field Documentation

- 6.257.3.1 `Pointer<DataStructure> activemq::commands::DataResponse::data`
[protected]
- 6.257.3.2 `const unsigned char activemq::commands::DataResponse::ID_ - DATARESPONSE = 32` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataResponse.h`

6.258 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1398).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.258.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1398). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.258.2 Constructor & Destructor Documentation

6.258.2.1 `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMarshaller()` [inline]

6.258.2.2 `virtual activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~~DataResponseMarshaller()` [inline, virtual]

6.258.3 Member Function Documentation

6.258.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.258.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.258.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.258.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.258.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.258.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2799).

6.258.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2800).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h**

6.259 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1402).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.259.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1402). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.259.2 Constructor & Destructor Documentation

6.259.2.1 `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMar
() [inline]`

6.259.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.259.3 Member Function Documentation

6.259.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.259.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.259.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2807).

6.259.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2808).

6.259.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2808).

6.259.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2809).

6.259.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2810).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h`

6.260 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1406).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.260.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1406). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.260.2 Constructor & Destructor Documentation

6.260.2.1 `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::DataResponseMar
() [inline]`

6.260.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.260.3 Member Function Documentation

6.260.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.260.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.260.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.260.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803).

6.260.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803).

6.260.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2804).

6.260.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataResponseMarshaller.h**

6.261 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1410).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.261.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1410). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.261.2 Constructor & Destructor Documentation

6.261.2.1 `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::DataResponseMar
() [inline]`

6.261.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]`

6.261.3 Member Function Documentation

6.261.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.261.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.261.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.261.3.4 virtual void `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.261.3.5 virtual int `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.261.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2814).

6.261.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2815).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h**

6.262 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1414).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual ~**DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.262.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1414). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.262.2 Constructor & Destructor Documentation

6.262.2.1 `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMarshaller()` [inline]

6.262.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~~DataResponseMarshaller()` [inline, virtual]

6.262.3 Member Function Documentation

6.262.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.262.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaller.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.262.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.262.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.262.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.262.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2794).

6.262.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2795).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataResponseMarshaller.h**

6.263 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference

Base class for all classes that marshal commands for Openwire.

#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::DataStreamMarshaller:

Public Member Functions

- virtual `~DataStreamMarshaller ()`
- virtual unsigned char `getDataStructureType ()` const =0
Gets the DataStructureType that this class marshals/unmarshals.
- virtual `commands::DataStructure * createObject ()` const =0
Creates a new instance of the class that this class is a marshaling director for.
- virtual int `tightMarshal (OpenWireFormat *format, commands::DataStructure *command, utils::BooleanStream *bs)=0` throw (decaf::io::IOException)
Tight Marshal to the given stream.
- virtual void `tightMarshal2 (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataOutputStream *ds, utils::BooleanStream *bs)=0` throw (decaf::io::IOException)
Tight Marshal to the given stream.
- virtual void `tightUnmarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataInputStream *dis, utils::BooleanStream *bs)=0` throw (decaf::io::IOException)
Tight Un-marshal to the given stream.
- virtual void `looseMarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataOutputStream *ds)=0` throw (decaf::io::IOException)
Tight Marshal to the given stream.
- virtual void `looseUnmarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataInputStream *dis)=0` throw (decaf::io::IOException)
Loose Un-marshal to the given stream.

6.263.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

6.263.2 Constructor & Destructor Documentation

6.263.2.1 `virtual`
`activemq::wireformat::openwire::marshal::DataStreamMarshaller::~~DataStreamMarshaller`
`() [inline, virtual]`

6.263.3 Member Function Documentation

6.263.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject`
`() const [pure virtual]`

Creates a new instance of the class that this class is a marshaling director for.

Returns:

newly allocated Command

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 182), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 218), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 327), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 391), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 428), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 484), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 558), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 583), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 607), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 760), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 788), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1145), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1169), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1196), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1226), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1314), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1343), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1368), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1532), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1684), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1785), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 1851), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 1868), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 1899), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 1923), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1950), `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 1966), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2010), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`

(p. 2211), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
(p. 2244), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
(p. 2270), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2301),
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2360), ac-
tivismq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2410),
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2490),
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2600),
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2624),
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2649),
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2708), ac-
tivismq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
(p. 2741), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
(p. 2773), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2807),
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2862), ac-
tivismq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2882), ac-
tivismq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2938), ac-
tivismq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3103),
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3246),
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3396),
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3420),
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
(p. 194), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
(p. 230), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
(p. 339), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
(p. 362), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
(p. 403), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
(p. 440), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
(p. 496), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
(p. 545), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
(p. 570), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
(p. 595), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
(p. 619), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 772),
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 800), ac-
tivismq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1157),
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1181),
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1208),
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1234),
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1272),
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1297),
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1326), ac-
tivismq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1351), ac-
tivismq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1376),
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1415),
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1490),
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1516), ac-
tivismq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1586),
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1680),
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1781),
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1839),
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1864),
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1887), ac-
tivismq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1911),
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1934),
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller

(p. 1962), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`
 (p. 1998), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`
 (p. 2195), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`
 (p. 2232), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`
 (p. 2258), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2285),
`activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2352), `ac-`
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2398),
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2478),
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2584),
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 2612),
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2637),
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2712), `ac-`
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`
 (p. 2737), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`
 (p. 2757), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792),
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 2866), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 2890), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 2950), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3119),
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3262),
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 3388),
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3428),
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`
 (p. 178), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`
 (p. 214), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`
 (p. 323), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`
 (p. 346), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`
 (p. 387), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`
 (p. 424), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`
 (p. 480), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`
 (p. 529), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`
 (p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`
 (p. 579), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`
 (p. 603), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 756),
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 784), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1141),
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1165),
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1192),
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1218),
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1256),
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1281),
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1310), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1335), `ac-`
`tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1360),
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1399),
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1494),
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1520), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1594),
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1688),
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1789),
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 1847),
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 1872),
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 1895), `ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 1915),

activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1942),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1970), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2002), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2203), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2240), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2266), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2293),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2356), ac-
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2402),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2482),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2588),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2616),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2653),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2716), ac-
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2745), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2761), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2797),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2870), ac-
 tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2898), ac-
 tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2946), ac-
 tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3111),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3254),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3380),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3416),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 186), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 222), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 331), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 354), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 395), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 432), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 488), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 537), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 587), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 611), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 764),
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 792), ac-
 tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1149),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1173),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1200),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1222),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1260),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1285),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1318), ac-
 tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1339), ac-
 tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1364),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1411),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1524), ac-
 tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1598),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1692),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1793),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 1843),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 1876),

activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 1891),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 1919),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1946),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller
 (p. 1974), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller
 (p. 2006), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller
 (p. 2207), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller
 (p. 2236), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2262), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2289),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2368),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2414),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2486),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2592),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2628),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2645),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2720),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 2749), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 2769), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2812),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 2874),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2886),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2942),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3107),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3250),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3392),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3424),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 190), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 226), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 335), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 358), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 399), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 436), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 492), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 541), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 566), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 591), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 615), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 768),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 796),
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1153),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1177),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1204),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1230),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1268),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1293),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1322),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1347),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1372),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1407),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1506),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1528),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1602),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1696),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1797),

activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 1855),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 1880),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 1903),
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 1927),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1938),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 1978), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 2014), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2199), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2248), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2274), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2297),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2364),
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2406),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2494),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2596),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2620),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2641),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2724),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2733), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2765), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2802),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 2858),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2894),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2954),
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3115),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3258),
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3384), and
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3432).

6.263.3.2 virtual unsigned char ac-

tivemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType
 () const [pure virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns:

byte Id of this classes DataStructureType

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller
 (p. 182), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller
 (p. 218), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller
 (p. 327), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller
 (p. 350), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller
 (p. 391), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller
 (p. 428), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller
 (p. 484), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller
 (p. 533), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller
 (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller
 (p. 583), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller
 (p. 607), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 760),
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 788),
 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1145),
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1169),

activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1196),
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1226),
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1264),
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1289),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1314),
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1343),
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1368),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1403),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1502),
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1532),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1590),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1684),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1785),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1851),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1868),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1899),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1923),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1950),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 1966),
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2010),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2211),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2244),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2270),
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2301),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2360),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2410),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2490),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2600),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2624),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2649),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2708),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2741),
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2773),
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2807),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2862),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2882),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2938),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3103),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3246),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3396),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3420),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 194),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 230),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 339),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 362),
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 403),
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 440),
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 496),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 545),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 570),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 595),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 619),
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 772),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 800), ac-

tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1157),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1181),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1208),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1234),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1272),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1297),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1326), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1351), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1376),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1415),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1516), ac-
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1586),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1680),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1781),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1839),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1864),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1887), ac-
 tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1911),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1934),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1962), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1998), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2195), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2232), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2258), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2285),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2352), ac-
 tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2398),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2478),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2584),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2612),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2637),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2712), ac-
 tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2737), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2757), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2792),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2866), ac-
 tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2890), ac-
 tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2950), ac-
 tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3119),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3262),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3388),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3428),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 178), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 214), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 323), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 346), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 387), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 424), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 480), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 529), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 579), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller

(p. 603), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 756),
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 784), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1141),
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1165),
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1192),
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1218),
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1256),
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1281),
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1310), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1335), `ac-`
`tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1360),
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1399),
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1494),
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1520), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1594),
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1688),
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1789),
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 1847),
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 1872),
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 1895), `ac-`
`tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 1915),
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 1942),
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`
(p. 1970), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`
(p. 2002), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`
(p. 2203), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`
(p. 2240), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`
(p. 2266), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2293),
`activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2356), `ac-`
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2402),
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2482),
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2588),
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 2616),
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2653),
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 2716), `ac-`
`tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`
(p. 2745), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`
(p. 2761), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797),
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 2870), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 2898), `ac-`
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 2946), `ac-`
`tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3111),
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3254),
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3380),
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3416),
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`
(p. 186), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`
(p. 222), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
(p. 331), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
(p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
(p. 395), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`
(p. 432), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
(p. 488), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
(p. 537), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`

(p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller (p. 587), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller (p. 611), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 764), activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 792), activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1149), activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1173), activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1200), activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1222), activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1260), activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1285), activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1318), activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1339), activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1364), activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1411), activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1498), activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1524), activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1598), activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1692), activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1793), activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 1843), activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 1876), activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 1891), activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 1919), activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1946), activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 1974), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2006), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2207), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2236), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2262), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2289), activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2368), activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2414), activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2486), activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2592), activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2628), activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2645), activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2720), activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 2749), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 2769), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2812), activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 2874), activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2886), activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2942), activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3107), activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3250), activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3392), activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3424), activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 190), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 226), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 335), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 358), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 399), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 436), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

(p. 492), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 541), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 591), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 615), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 768), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 796), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1153), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1177), `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1204), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1230), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1268), `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1293), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1322), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1347), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1506), `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1528), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1602), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1696), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1797), `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1855), `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1880), `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1903), `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1927), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1938), `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1978), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2014), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2199), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2248), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2274), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2297), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2364), `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2406), `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2494), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2596), `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2620), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2641), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2724), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 2733), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 2765), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802), `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2858), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2894), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2954), `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3115), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3258), `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3384), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3432).

6.263.3.3 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) throw (decaf::io::IOException)
[pure virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenWireFormat properties

command - the object to Marshal

ds - DataOutputStream to marshal to

Exceptions:

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 182), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 218), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 290), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 327), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 350), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 391), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 428), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 484), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 508), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 533), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 558), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 583), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 607), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 665), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 760), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 788), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1145), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1169), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1196), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1226), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1264), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1289), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1314), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1343), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1368), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1403), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1502), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1532), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1590), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1684), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1785), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1851), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1868), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1899), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1923), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1950),

activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1966), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2010), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2211), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2244), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2270), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2301),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2326), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2360), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2410),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2490),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2600),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2624),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2649),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2708), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2741), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2773), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2807),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2862), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2882), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2938), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3103),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3221),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3246),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3396),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3420),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 194), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 230), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 302), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 339), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 362), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 403), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 440), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 496), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 520), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 545), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 570), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 595), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 619), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 686), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 772),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 800), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1157),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1181),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1208),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1234),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1272),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1297),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1326), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1351), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1376),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1415),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1516), ac-
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1586),

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1680),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1781),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1839),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1864),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1887),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1911),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1934),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1962), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1998), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2195), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2232), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2258), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2285),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2306),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2352),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2398),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2478),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2584),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2612),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2637),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2712),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2737), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2757), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2792),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2866),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2890),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2950),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3119),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3225),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3262),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3388),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3428),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 178), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 214), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 286), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 323), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 346), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 387), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 424), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 480), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 504), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 529), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 579), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 603), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 658), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 756),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 784),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1141),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1165),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1192),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1218),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1256),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1281),

activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1310),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1335),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1360),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1399),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1494),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1520),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1594),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1688),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1789),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1847),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1872),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1895),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1915),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1942),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1970), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2002), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2203), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2240), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2266), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2293),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2316),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2356),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2402),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2482),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2588),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2616),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2653),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2716),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2745), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2761), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2797),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2870),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2898),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2946),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3111),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3237),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3254),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3380),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3416),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 186), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 222), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 294), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 331), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 354), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 395), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 432), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 488), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 512), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 537), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 562), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 587), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 611), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller
 (p. 672), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 764),

activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 792),
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1149),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1173),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1200),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1222),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1260),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1285),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1318),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1339),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1364),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1411),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1524),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1598),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1692),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1793),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 1843),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 1876),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 1891),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 1919),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1946),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 1974),
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2006),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2207),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2236),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2262),
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2289),
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2311),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2368),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2414),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2486),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2592),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2628),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2645),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2720),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 2749),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 2769),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2812),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 2874),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2886),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2942),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3107),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3229),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3250),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3392),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3424),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 190),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 226),
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 298),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 335),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 358),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 399),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 436),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

(p. 492), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 516), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 541), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 566), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 591), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 615), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 679), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 768),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 796), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1153),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1177),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1204),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1230),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1268),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1293),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1322), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1347), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1372),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1407),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1506),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1528), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1602),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1696),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1797),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1855),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1880),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1903), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1927),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1938),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1978), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2014), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2199), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2248), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2274), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2297),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2364), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2406),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2494),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2596),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2620),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2641),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2724), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2733), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2765), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2858), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2894), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2954), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3115),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3233), `ac-`
`tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3258), `ac-`
`tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3384), and
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3432).

6.263.3.4 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException) [pure virtual]

Loose Un-marhsal to the given stream.

Parameters:

format - The OpenWireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from

Exceptions:

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 183), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 219), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 290), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 328), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 351), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 392), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 429), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 485), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 508), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 534), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 559), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 584), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 608), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 666), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 761), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 789), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1146), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1170), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1197), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1227), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1265), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1290), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1315), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1344), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1369), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1404), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1503), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1533), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1591), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1685), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1786), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1852), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1869), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1900), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1924), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1951),

activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1967), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2011), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2212), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2245), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2271), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2302),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2326), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2361), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2411),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2491),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2601),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2625),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2650),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2709), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2742), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2774), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2808),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2863), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2883), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2939), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3104),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3221),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3247),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3397),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3421),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 195), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 231), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 302), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 340), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 363), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 404), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 441), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 497), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 520), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 546), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 596), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 620), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 687), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 773),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 801), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1158),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1182),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1209),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1235),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1298),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1327), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1352), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1377),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1416),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1491),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1517), ac-
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1587),

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1782),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1840),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1865),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1888),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1912),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1935),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1963), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1999), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2196), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2233), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2259), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2286),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2306),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2353),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2399),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2479),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2585),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2613),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2638),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2713),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2738), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2758), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2793),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2867),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2891),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2951),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3120),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3225),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3263),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3389),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3429),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 179), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 215), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 286), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 324), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 347), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 388), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 425), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 481), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 504), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 530), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 580), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 604), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 659), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 757),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 785),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1142),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1166),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1193),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1219),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1257),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1282),

activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1311),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1336),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1361),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1400),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1521),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1595),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1790),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1848),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1873),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1896),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1916),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1943),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1971), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2003), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2204), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2241), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2267), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2294),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2316),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2357),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2403),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2483),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2617),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2654),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2717),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2746), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2762), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2798),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2871),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2899),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2947),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3112),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3237),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3255),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3381),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3417),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 187), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 223), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 294), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 332), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 355), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 396), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 433), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 489), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 512), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 538), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 563), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 588), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 612), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller
 (p. 673), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 765),

activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 793),
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1150),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1174),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1201),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1223),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1261),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1286),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1319),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1340),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1365),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1499),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1525),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1599),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1693),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1794),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 1844),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 1877),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 1892),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 1920),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1947),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 1975),
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2007),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2208),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2237),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2263),
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2290),
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2311),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2369),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2415),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2487),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2593),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2629),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2646),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2721),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 2750),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 2770),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2813),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 2875),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2887),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2943),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3108),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3229),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3251),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3393),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3425),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 191),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 227),
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 298),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 336),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 359),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 400),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 437),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

(p. 493), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 516), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 542), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 592), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 616), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 680), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 769),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 797), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1154),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1178),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1205),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1231),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1269),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1294),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1323), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1348), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1373),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1408),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1507),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1529), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1603),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1697),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1798),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1856),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1881),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1904), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1928),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1939),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1979), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2015), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2200), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2249), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2275), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2298),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2321), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2365), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2407),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2495),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2597),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2621),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2642),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2725), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2734), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2766), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2859), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2895), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2955), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3116),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3233), `ac-`
`tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3259), `ac-`
`tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3385), and
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3433).

```

6.263.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1
(OpenWireFormat * format, commands::DataStructure * command,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [pure
virtual]

```

Tight Marshal to the given stream.

Parameters:

format - The OpenWireFormat properties

command - the object to Marshal

bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 183), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 291), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 328), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 429), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 485), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 509), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 534), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 584), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 608), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 761), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 789), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1146), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1170), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1197), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1227), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1265), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1290), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1344), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1369), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1404), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1533), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1685), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1786), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 1852), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 1869), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 1900), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 1924), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1951),

activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1967), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2011), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2212), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2245), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2271), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2302),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2327), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2361), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2411),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2491),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2601),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2625),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2650),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2709), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2742), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2774), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2808),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2863), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2883), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2939), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3104),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3222),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3247),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3397),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3421),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 195), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 231), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 303), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 340), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 363), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 404), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 441), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 497), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 521), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 546), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 596), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 620), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 688), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 773),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 801), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1158),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1182),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1209),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1235),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1298),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1327), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1352), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1377),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1416),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1491),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1517), ac-
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1587),

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1782),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1840),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1865),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1888),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1912),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1935),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1963), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1999), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2196), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2233), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2259), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2286),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2307),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2353),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2399),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2479),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2585),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2613),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2638),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2713),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2738), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2758), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2793),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2867),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2891),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2951),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3120),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3226),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3263),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3389),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3429),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 179), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 215), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 287), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 324), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 347), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 388), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 425), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 481), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 505), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 530), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 580), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 604), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 660), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 757),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 785),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1142),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1166),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1193),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1219),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1257),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1282),

activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1311),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1336),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1361),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1400),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1521),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1595),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1790),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1848),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1873),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1896),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1916),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1943),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1971), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2003), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2204), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2241), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2267), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2294),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2317),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2357),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2403),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2483),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2617),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2654),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2717),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2746), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2762), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2798),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2871),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2899),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2947),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3112),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3238),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3255),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3381),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3417),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 187), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 223), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 295), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 332), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 355), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 396), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 433), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 489), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 513), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 538), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 563), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 588), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 612), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller
 (p. 674), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 765),

activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 793),
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1150),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1174),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1201),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1223),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1261),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1286),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1319),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1340),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1365),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1499),
 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1525),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1599),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1693),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1794),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 1844),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 1877),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 1892),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 1920),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1947),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 1975),
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2007),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2208),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2237),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2263),
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2290),
 activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2312),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2369),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2415),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2487),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2593),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2629),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2646),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2721),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 2750),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 2770),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2813),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 2875),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2887),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2943),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3108),
 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3230),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3251),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3393),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3425),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 191),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 227),
 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 299),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 336),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 359),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 400),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 437),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

(p. 493), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 517), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 542), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 592), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 616), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 681), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 769),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 797), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1154),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1178),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1205),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1231),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1269),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1294),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1323), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1348), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1373),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1408),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1507),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1529), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1603),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1697),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1798),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1856),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1881),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1904), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1928),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1939),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1979), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2015), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2200), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2249), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2275), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2298),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2322), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2365), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2407),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2495),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2597),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2621),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2642),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2725), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2734), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2766), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2859), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2895), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2955), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3116),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3234), `ac-`
`tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3259), `ac-`
`tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3385), and
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3433).

6.263.3.6 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [pure virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenWireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to marshal to.

Exceptions:

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 183), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 219), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 291), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 328), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 351), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 392), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 429), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 485), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 509), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 534), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 559), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 584), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 608), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 668), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 761), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 789), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1146), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1170), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1197), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1227), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1265), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1290), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1315), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1344), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1370), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1405), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1503), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1533), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1592), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1685), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1787), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1852), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1869), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1900), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1924),

activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1951),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1968), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2011), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2212), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2245), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2271), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2302),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2328), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2361), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2411),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2601),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2625),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2650),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2709), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2742), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2774), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2809),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2863), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2883), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2939), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3104),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3222),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3247),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3397),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3421),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 195), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 231), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 303), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 340), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 363), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 404), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 441), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 497), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 521), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 546), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 596), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 620), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 689), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 773),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 801), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1158),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1182),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1209),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1235),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1298),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1327), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1352), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1378),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1417),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1491),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1517), ac-

tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1588),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1783),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1840),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1865),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1888),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1912),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1935),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1964), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1999), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2196), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2233), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2259), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2286),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2308),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2353),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2399),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2480),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2585),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2613),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2638),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2713),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2738), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2758), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2794),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2867),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2891),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2951),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3120),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3226),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3263),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3389),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3429),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 179), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 215), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 287), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 324), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 347), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 388), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 425), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 481), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 505), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 530), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 580), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 604), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 661), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 757),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 785),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1142),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1166),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1193),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1219),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1257),

activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1282),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1311),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1336),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1362),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1401),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1495),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1521),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1596),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1791),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1848),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1873),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1896),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1916),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1943),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1972), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2003), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2204), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2241), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2267), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2294),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2318),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2357),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2403),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2484),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2617),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2654),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2717),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2746), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2762), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2799),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2871),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2899),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2947),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3112),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3238),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3255),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3381),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3417),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 187), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 223), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 295), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 332), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 355), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 396), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 433), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 489), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 513), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 538), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 563), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 588), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 612), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller

(p. 675), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 765),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 793), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1150),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1174),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1201),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1223),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1261),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1286),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1319), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1340), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1366),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1413),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1499),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1525), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1600),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1693),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1795),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1844),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1877),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1892), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1920),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1947),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
(p. 1976), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
(p. 2007), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
(p. 2208), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
(p. 2237), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
(p. 2263), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2290),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2369), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2415),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2488),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2593),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2629),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2646),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2721), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
(p. 2750), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
(p. 2770), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2814),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2875), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2887), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2943), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3108),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3230),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3251),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3393),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3425),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
(p. 191), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
(p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`
(p. 299), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
(p. 336), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
(p. 359), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
(p. 400), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`

(p. 437), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 493), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 517), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 542), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 592), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 616), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 682), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 769),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 797), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1154),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1178),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1205),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1231),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1269),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1294),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1323), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1348), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1374),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1409),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1507),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1529), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1604),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1697),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1799),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1856),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1881),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1904), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1928),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1939),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1980), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2015), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2200), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2249), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2275), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2298),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2365), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2407),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2496),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2597),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2621),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2642),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2725), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2734), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2766), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2804),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2859), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2895), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2955), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3116),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3234), `ac-`
`tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3259), `ac-`
`tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3385), and
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3433).

6.263.3.7 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [pure virtual]

Tight Un-marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from
bs - boolean stream to unmarshal from.

Exceptions:

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 184), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 220), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 292), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 329), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 352), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 393), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 430), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 486), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 510), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 535), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 560), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 585), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 609), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 669), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 762), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 790), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1147), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1171), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1198), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1228), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1266), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1291), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1316), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1345), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1370), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1405), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1504), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1534), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1592), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1686), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1787), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1853), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1870), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1901), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1925),

activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1952),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1968), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 2012), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2213), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2246), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2272), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2303),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2328), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2362), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2412),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2602),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2626),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2651),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2710), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2743), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2775), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2810),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2864), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2884), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2940), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3105),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3223),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3248),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3398),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3422),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 196), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 232), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 304), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 341), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 364), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 405), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 442), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 498), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 522), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 547), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 572), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 597), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 621), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 690), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 774),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 802), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1159),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1183),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1210),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1236),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1274),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1299),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1328), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1353), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1378),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1417),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1492),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1518), ac-

tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1588),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1682),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1783),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1841),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1866),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1889),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1913),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1936),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1964), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 2000), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2197), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2234), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2260), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2287),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2308),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2354),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2400),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2480),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2586),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2614),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2639),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2714),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2739), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2759), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2795),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2868),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2892),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2952),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3121),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3227),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3264),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3390),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3430),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 180), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 216), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 288), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 325), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 348), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 389), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 426), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 482), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 506), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 531), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 556), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 581), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 605), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 662), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 758),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 786),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1143),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1167),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1194),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1220),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1258),

activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1283),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1312),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1337),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1362),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1401),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1496),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1522),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1596),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1690),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1791),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1849),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1874),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1897),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1917),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1944),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1972), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 2004), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2205), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2242), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2268), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2295),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2318),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2358),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2404),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2484),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2590),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2618),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2655),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2718),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2747), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2763), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2800),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2872),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2900),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2948),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3113),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3239),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3256),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3382),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3418),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 188), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 224), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 296), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 333), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 356), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 397), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 434), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 490), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 514), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 539), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 564), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 589), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 613), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller

(p. 676), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 766),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 794), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1151),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1175),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1202),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1224),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1262),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1287),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1320), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1341), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1366),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1413),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1500),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1526), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1600),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1694),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1795),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1845),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1878),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1893), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1921),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1948),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
(p. 1976), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
(p. 2008), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
(p. 2209), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
(p. 2238), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
(p. 2264), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2291),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2313), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2370), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2416),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2488),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2594),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2630),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2647),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2722), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
(p. 2751), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
(p. 2771), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2815),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2876), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2888), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2944), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3109),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3231),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3252),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3394),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3426),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
(p. 192), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
(p. 228), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`
(p. 300), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
(p. 337), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
(p. 360), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
(p. 401), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`

(p. 438), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 494), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 518), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 543), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 568), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 593), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 617), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 683), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 770),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 798), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1155),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1179),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1206),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1232),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1270),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1295),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1324), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1349), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1374),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1409),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1508),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1530), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1604),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1698),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1799),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1857),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1882),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1905), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1929),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1940),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1980), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 2016), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2201), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2250), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2276), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2299),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2323), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2366), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2408),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2496),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2598),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2622),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2643),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2726), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2735), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2767), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2805),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2860), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2896), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2956), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3117),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3235), `ac-`
`tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3260), `ac-`
`tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3386), and
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3434).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h`

6.264 activemq::commands::DataStructure Class Reference

#include <src/main/activemq/commands/DataStructure.h> Inheritance diagram for activemq::commands::DataStructure:

Public Member Functions

- virtual **~DataStructure** ()
- virtual unsigned char **getDataStructureType** () const =0
*Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.*
- virtual **DataStructure * cloneDataStructure** () const =0
Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)=0
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const =0
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const =0
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*

6.264.1 Constructor & Destructor Documentation

6.264.1.1 virtual activemq::commands::DataStructure::~~DataStructure ()
 [inline, virtual]

6.264.2 Member Function Documentation

6.264.2.1 virtual **DataStructure*** activemq::commands::DataStructure::cloneDataStructure ()
 const [pure virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 173), **activemq::commands::ActiveMQBytesMessage** (p. 200), **activemq::commands::ActiveMQDestination** (p. 276), **activemq::commands::ActiveMQMapMessage** (p. 312), **activemq::commands::ActiveMQMessage** (p. 343), **activemq::commands::ActiveMQObjectMessage**

(p. 384), `activemq::commands::ActiveMQQueue` (p. 420), `ac-`
`tivemq::commands::ActiveMQStreamMessage` (p. 467), `ac-`
`tivemq::commands::ActiveMQTempDestination` (p. 500), `ac-`
`tivemq::commands::ActiveMQTempQueue` (p. 524), `ac-`
`tivemq::commands::ActiveMQTempTopic` (p. 549), `activemq::commands::ActiveMQTextMessage`
(p. 575), `activemq::commands::ActiveMQTopic` (p. 599), `ac-`
`tivemq::commands::BooleanExpression` (p. 737), `activemq::commands::BrokerError`
(p. 746), `activemq::commands::BrokerId` (p. 752), `activemq::commands::BrokerInfo`
(p. 777), `activemq::commands::ConnectionControl` (p. 1136), `ac-`
`tivemq::commands::ConnectionError` (p. 1161), `activemq::commands::ConnectionId`
(p. 1188), `activemq::commands::ConnectionInfo` (p. 1212), `ac-`
`tivemq::commands::ConsumerControl` (p. 1251), `activemq::commands::ConsumerId`
(p. 1276), `activemq::commands::ConsumerInfo` (p. 1302), `ac-`
`tivemq::commands::ControlCommand` (p. 1331), `activemq::commands::DataArrayResponse`
(p. 1357), `activemq::commands::DataResponse` (p. 1396), `ac-`
`tivemq::commands::DestinationInfo` (p. 1485), `activemq::commands::DiscoveryEvent`
(p. 1512), `activemq::commands::ExceptionResponse` (p. 1583), `ac-`
`tivemq::commands::FlushCommand` (p. 1677), `activemq::commands::IntegerResponse`
(p. 1778), `activemq::commands::JournalQueueAck` (p. 1835), `ac-`
`tivemq::commands::JournalTopicAck` (p. 1859), `activemq::commands::JournalTrace`
(p. 1884), `activemq::commands::JournalTransaction` (p. 1907), `ac-`
`tivemq::commands::KeepAliveInfo` (p. 1931), `activemq::commands::LastPartialCommand`
(p. 1959), `activemq::commands::LocalTransactionId` (p. 1994), `ac-`
`tivemq::commands::Message` (p. 2149), `activemq::commands::MessageAck`
(p. 2189), `activemq::commands::MessageDispatch` (p. 2220), `ac-`
`tivemq::commands::MessageDispatchNotification` (p. 2252), `ac-`
`tivemq::commands::MessageId` (p. 2280), `activemq::commands::MessagePull`
(p. 2347), `activemq::commands::NetworkBridgeFilter` (p. 2394), `ac-`
`tivemq::commands::PartialCommand` (p. 2474), `activemq::commands::ProducerAck`
(p. 2580), `activemq::commands::ProducerId` (p. 2607), `ac-`
`tivemq::commands::ProducerInfo` (p. 2632), `activemq::commands::RemoveInfo`
(p. 2704), `activemq::commands::RemoveSubscriptionInfo` (p. 2728), `ac-`
`tivemq::commands::ReplayCommand` (p. 2753), `activemq::commands::Response`
(p. 2782), `activemq::commands::SessionId` (p. 2854), `ac-`
`tivemq::commands::SessionInfo` (p. 2878), `activemq::commands::ShutdownInfo`
(p. 2935), `activemq::commands::SubscriptionInfo` (p. 3098), `ac-`
`tivemq::commands::TransactionId` (p. 3218), `activemq::commands::TransactionInfo`
(p. 3241), `activemq::commands::WireFormatInfo` (p. 3372), and `ac-`
`tivemq::commands::XATransactionId` (p. 3411).

6.264.2.2 virtual void `activemq::commands::DataStructure::copyDataStructure` (const `DataStructure * src`) [pure virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

`src` - Source Object

Implemented in `activemq::commands::ActiveMQBlobMessage`
(p. 173), `activemq::commands::ActiveMQBytesMessage` (p. 201),
`activemq::commands::ActiveMQDestination` (p. 277), `ac-`
`tivemq::commands::ActiveMQMapMessage` (p. 312), `ac-`
`tivemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQObjectMessage`

(p. 384), `activemq::commands::ActiveMQQueue` (p. 420), `ac-`
`tivemq::commands::ActiveMQStreamMessage` (p. 467), `ac-`
`tivemq::commands::ActiveMQTempDestination` (p. 500), `ac-`
`tivemq::commands::ActiveMQTempQueue` (p. 525), `ac-`
`tivemq::commands::ActiveMQTempTopic` (p. 550), `activemq::commands::ActiveMQTextMessage`
(p. 575), `activemq::commands::ActiveMQTopic` (p. 599), `ac-`
`tivemq::commands::BaseCommand` (p. 651), `activemq::commands::BrokerError`
(p. 746), `activemq::commands::BrokerId` (p. 752), `activemq::commands::BrokerInfo`
(p. 777), `activemq::commands::ConnectionControl` (p. 1136), `ac-`
`tivemq::commands::ConnectionError` (p. 1161), `activemq::commands::ConnectionId`
(p. 1188), `activemq::commands::ConnectionInfo` (p. 1213), `ac-`
`tivemq::commands::ConsumerControl` (p. 1251), `activemq::commands::ConsumerId`
(p. 1276), `activemq::commands::ConsumerInfo` (p. 1302), `ac-`
`tivemq::commands::ControlCommand` (p. 1331), `activemq::commands::DataArrayResponse`
(p. 1357), `activemq::commands::DataResponse` (p. 1396), `ac-`
`tivemq::commands::DestinationInfo` (p. 1485), `activemq::commands::DiscoveryEvent`
(p. 1512), `activemq::commands::ExceptionResponse` (p. 1583), `ac-`
`tivemq::commands::FlushCommand` (p. 1677), `activemq::commands::IntegerResponse`
(p. 1778), `activemq::commands::JournalQueueAck` (p. 1835), `ac-`
`tivemq::commands::JournalTopicAck` (p. 1859), `activemq::commands::JournalTrace`
(p. 1884), `activemq::commands::JournalTransaction` (p. 1907), `ac-`
`tivemq::commands::KeepAliveInfo` (p. 1931), `activemq::commands::LastPartialCommand`
(p. 1959), `activemq::commands::LocalTransactionId` (p. 1994), `ac-`
`tivemq::commands::Message` (p. 2150), `activemq::commands::MessageAck`
(p. 2189), `activemq::commands::MessageDispatch` (p. 2220), `ac-`
`tivemq::commands::MessageDispatchNotification` (p. 2252), `ac-`
`tivemq::commands::MessageId` (p. 2280), `activemq::commands::MessagePull`
(p. 2347), `activemq::commands::NetworkBridgeFilter` (p. 2394), `ac-`
`tivemq::commands::PartialCommand` (p. 2474), `activemq::commands::ProducerAck`
(p. 2580), `activemq::commands::ProducerId` (p. 2607), `ac-`
`tivemq::commands::ProducerInfo` (p. 2632), `activemq::commands::RemoveInfo`
(p. 2704), `activemq::commands::RemoveSubscriptionInfo` (p. 2728), `ac-`
`tivemq::commands::ReplayCommand` (p. 2753), `activemq::commands::Response`
(p. 2782), `activemq::commands::SessionId` (p. 2854), `ac-`
`tivemq::commands::SessionInfo` (p. 2878), `activemq::commands::ShutdownInfo`
(p. 2935), `activemq::commands::SubscriptionInfo` (p. 3098), `ac-`
`tivemq::commands::TransactionId` (p. 3218), `activemq::commands::TransactionInfo`
(p. 3241), `activemq::commands::WireFormatInfo` (p. 3372), and `ac-`
`tivemq::commands::XATransactionId` (p. 3411).

6.264.2.3 `virtual bool activemq::commands::DataStructure::equals (const DataStructure * value) const` [pure virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Implemented in `activemq::commands::ActiveMQBlobMessage`
(p. 174), `activemq::commands::ActiveMQBytesMessage` (p. 201),
`activemq::commands::ActiveMQDestination` (p. 278), `ac-`

tivemq::commands::ActiveMQMapMessage (p. 312), ac-
 tivemq::commands::ActiveMQMessage (p. 343), activemq::commands::ActiveMQMessageTemplate<
 T > (p. 369), activemq::commands::ActiveMQObjectMessage
 (p. 384), activemq::commands::ActiveMQQueue (p. 421), ac-
 tivemq::commands::ActiveMQStreamMessage (p. 468), ac-
 tivemq::commands::ActiveMQTempDestination (p. 501), ac-
 tivemq::commands::ActiveMQTempQueue (p. 525), ac-
 tivemq::commands::ActiveMQTempTopic (p. 550), activemq::commands::ActiveMQTextMessage
 (p. 575), activemq::commands::ActiveMQTopic (p. 600), ac-
 tivemq::commands::BaseCommand (p. 651), activemq::commands::BooleanExpression
 (p. 738), activemq::commands::BrokerId (p. 752), activemq::commands::BrokerInfo
 (p. 777), activemq::commands::ConnectionControl (p. 1137), ac-
 tivemq::commands::ConnectionError (p. 1161), activemq::commands::ConnectionId
 (p. 1189), activemq::commands::ConnectionInfo (p. 1213), ac-
 tivemq::commands::ConsumerControl (p. 1252), activemq::commands::ConsumerId
 (p. 1277), activemq::commands::ConsumerInfo (p. 1303), ac-
 tivemq::commands::ControlCommand (p. 1331), activemq::commands::DataArrayResponse
 (p. 1357), activemq::commands::DataResponse (p. 1396), ac-
 tivemq::commands::DestinationInfo (p. 1486), activemq::commands::DiscoveryEvent
 (p. 1512), activemq::commands::ExceptionResponse (p. 1583), ac-
 tivemq::commands::FlushCommand (p. 1677), activemq::commands::IntegerResponse
 (p. 1778), activemq::commands::JournalQueueAck (p. 1835), ac-
 tivemq::commands::JournalTopicAck (p. 1860), activemq::commands::JournalTrace
 (p. 1884), activemq::commands::JournalTransaction (p. 1907), ac-
 tivemq::commands::KeepAliveInfo (p. 1931), activemq::commands::LastPartialCommand
 (p. 1959), activemq::commands::LocalTransactionId (p. 1995), ac-
 tivemq::commands::Message (p. 2150), activemq::commands::MessageAck
 (p. 2190), activemq::commands::MessageDispatch (p. 2221), ac-
 tivemq::commands::MessageDispatchNotification (p. 2253), ac-
 tivemq::commands::MessageId (p. 2281), activemq::commands::MessagePull
 (p. 2348), activemq::commands::NetworkBridgeFilter (p. 2394), ac-
 tivemq::commands::PartialCommand (p. 2474), activemq::commands::ProducerAck
 (p. 2580), activemq::commands::ProducerId (p. 2608), ac-
 tivemq::commands::ProducerInfo (p. 2633), activemq::commands::RemoveInfo
 (p. 2704), activemq::commands::RemoveSubscriptionInfo (p. 2729), ac-
 tivemq::commands::ReplayCommand (p. 2753), activemq::commands::Response
 (p. 2782), activemq::commands::SessionId (p. 2855), ac-
 tivemq::commands::SessionInfo (p. 2878), activemq::commands::ShutdownInfo
 (p. 2935), activemq::commands::SubscriptionInfo (p. 3099), ac-
 tivemq::commands::TransactionId (p. 3218), activemq::commands::TransactionInfo
 (p. 3241), activemq::commands::WireFormatInfo (p. 3372), ac-
 tivemq::commands::XATransactionId (p. 3412), activemq::commands::ActiveMQMessageTemplate<
 cms::BytesMessage > (p. 369), activemq::commands::ActiveMQMessageTemplate<
 cms::MapMessage > (p. 369), activemq::commands::ActiveMQMessageTemplate<
 cms::Message > (p. 369), activemq::commands::ActiveMQMessageTemplate<
 cms::StreamMessage > (p. 369), activemq::commands::ActiveMQMessageTemplate<
 cms::TextMessage > (p. 369), and activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage > (p. 369).

6.264.2.4 virtual unsigned char activemq::commands::DataStructure::getDataStructureType
 () const [pure virtual]

Get the **DataStructure** (p. 1461) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 174), **activemq::commands::ActiveMQBytesMessage** (p. 202), **activemq::commands::ActiveMQDestination** (p. 278), **activemq::commands::ActiveMQMapMessage** (p. 314), **activemq::commands::ActiveMQMessage** (p. 343), **activemq::commands::ActiveMQObjectMessage** (p. 385), **activemq::commands::ActiveMQQueue** (p. 421), **activemq::commands::ActiveMQStreamMessage** (p. 468), **activemq::commands::ActiveMQTempDestination** (p. 501), **activemq::commands::ActiveMQTempQueue** (p. 526), **activemq::commands::ActiveMQTempTopic** (p. 551), **activemq::commands::ActiveMQTextMessage** (p. 575), **activemq::commands::ActiveMQTopic** (p. 600), **activemq::commands::BrokerError** (p. 747), **activemq::commands::BrokerId** (p. 753), **activemq::commands::BrokerInfo** (p. 778), **activemq::commands::ConnectionControl** (p. 1137), **activemq::commands::ConnectionError** (p. 1162), **activemq::commands::ConnectionId** (p. 1189), **activemq::commands::ConnectionInfo** (p. 1213), **activemq::commands::ConsumerControl** (p. 1252), **activemq::commands::ConsumerId** (p. 1277), **activemq::commands::ConsumerInfo** (p. 1303), **activemq::commands::ControlCommand** (p. 1332), **activemq::commands::DataArrayResponse** (p. 1358), **activemq::commands::DataResponse** (p. 1397), **activemq::commands::DestinationInfo** (p. 1486), **activemq::commands::DiscoveryEvent** (p. 1513), **activemq::commands::ExceptionResponse** (p. 1583), **activemq::commands::FlushCommand** (p. 1677), **activemq::commands::IntegerResponse** (p. 1778), **activemq::commands::JournalQueueAck** (p. 1835), **activemq::commands::JournalTopicAck** (p. 1860), **activemq::commands::JournalTrace** (p. 1884), **activemq::commands::JournalTransaction** (p. 1908), **activemq::commands::KeepAliveInfo** (p. 1931), **activemq::commands::LastPartialCommand** (p. 1960), **activemq::commands::LocalTransactionId** (p. 1995), **activemq::commands::Message** (p. 2152), **activemq::commands::MessageAck** (p. 2190), **activemq::commands::MessageDispatch** (p. 2221), **activemq::commands::MessageDispatchNotification** (p. 2253), **activemq::commands::MessageId** (p. 2281), **activemq::commands::MessagePull** (p. 2348), **activemq::commands::NetworkBridgeFilter** (p. 2395), **activemq::commands::PartialCommand** (p. 2475), **activemq::commands::ProducerAck** (p. 2581), **activemq::commands::ProducerId** (p. 2608), **activemq::commands::ProducerInfo** (p. 2633), **activemq::commands::RemoveInfo** (p. 2705), **activemq::commands::RemoveSubscriptionInfo** (p. 2729), **activemq::commands::ReplayCommand** (p. 2754), **activemq::commands::Response** (p. 2783), **activemq::commands::SessionId** (p. 2855), **activemq::commands::SessionInfo** (p. 2879), **activemq::commands::ShutdownInfo** (p. 2935), **activemq::commands::SubscriptionInfo** (p. 3099), **activemq::commands::TransactionId** (p. 3219), **activemq::commands::TransactionInfo** (p. 3242), **activemq::commands::WireFormatInfo** (p. 3373), and **activemq::commands::XATransactionId** (p. 3412).

6.264.2.5 virtual std::string activemq::commands::DataStructure::toString () const [pure virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p.176), **activemq::commands::ActiveMQBytesMessage** (p.208), **activemq::commands::ActiveMQDestination** (p.282), **activemq::commands::ActiveMQMapMessage** (p.320), **activemq::commands::ActiveMQMessage** (p.344), **activemq::commands::ActiveMQObjectMessage** (p.385), **activemq::commands::ActiveMQQueue** (p.422), **activemq::commands::ActiveMQStreamMessage** (p.474), **activemq::commands::ActiveMQTempDestination** (p.501), **activemq::commands::ActiveMQTempQueue** (p.526), **activemq::commands::ActiveMQTempTopic** (p.551), **activemq::commands::ActiveMQTextMessage** (p.577), **activemq::commands::ActiveMQTopic** (p.601), **activemq::commands::BaseCommand** (p.655), **activemq::commands::BaseDataStructure** (p.718), **activemq::commands::BooleanExpression** (p.738), **activemq::commands::BrokerId** (p.753), **activemq::commands::BrokerInfo** (p.780), **activemq::commands::Command** (p.1067), **activemq::commands::ConnectionControl** (p.1138), **activemq::commands::ConnectionError** (p.1162), **activemq::commands::ConnectionId** (p.1189), **activemq::commands::ConnectionInfo** (p.1215), **activemq::commands::ConsumerControl** (p.1253), **activemq::commands::ConsumerId** (p.1278), **activemq::commands::ConsumerInfo** (p.1305), **activemq::commands::ControlCommand** (p.1332), **activemq::commands::DataArrayResponse** (p.1358), **activemq::commands::DataResponse** (p.1397), **activemq::commands::DestinationInfo** (p.1487), **activemq::commands::DiscoveryEvent** (p.1513), **activemq::commands::ExceptionResponse** (p.1584), **activemq::commands::FlushCommand** (p.1678), **activemq::commands::IntegerResponse** (p.1779), **activemq::commands::JournalQueueAck** (p.1836), **activemq::commands::JournalTopicAck** (p.1861), **activemq::commands::JournalTrace** (p.1885), **activemq::commands::JournalTransaction** (p.1908), **activemq::commands::KeepAliveInfo** (p.1932), **activemq::commands::LastPartialCommand** (p.1960), **activemq::commands::LocalTransactionId** (p.1996), **activemq::commands::Message** (p.2159), **activemq::commands::MessageAck** (p.2192), **activemq::commands::MessageDispatch** (p.2222), **activemq::commands::MessageDispatchNotification** (p.2255), **activemq::commands::MessageId** (p.2282), **activemq::commands::MessagePull** (p.2349), **activemq::commands::NetworkBridgeFilter** (p.2395), **activemq::commands::PartialCommand** (p.2475), **activemq::commands::ProducerAck** (p.2581), **activemq::commands::ProducerId** (p.2609), **activemq::commands::ProducerInfo** (p.2634), **activemq::commands::RemoveInfo** (p.2705), **activemq::commands::RemoveSubscriptionInfo** (p.2730), **activemq::commands::ReplayCommand** (p.2754), **activemq::commands::Response** (p.2783), **activemq::commands::SessionId** (p.2856), **activemq::commands::SessionInfo** (p.2879), **activemq::commands::ShutdownInfo** (p.2936), **activemq::commands::SubscriptionInfo** (p.3100), **activemq::commands::TransactionId** (p.3219), **activemq::commands::TransactionInfo**

(p. 3243), `activemq::commands::WireFormatInfo` (p. 3378), and `activemq::commands::XATransactionId` (p. 3413).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataSetStructure.h`

6.265 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

#include <src/main/decaf/util/Date.h> Inheritance diagram for decaf::util::Date:

Public Member Functions

- **Date** ()
Default constructor - sets time to the current System time, rounded to the nearest millisecond.
- **Date** (long long milliseconds)
Constructs the date with a given time value.
- **Date** (const **Date** &source)
Copy constructor.
- **Date** & **operator=** (const **Date** &value)
*Assigns the value of one **Date** (p. 1467) object to another.*
- virtual ~**Date** ()
- long long **getTime** () const
Gets the underlying time.
- void **setTime** (long long milliseconds)
Sets the underlying time.
- bool **after** (const **Date** &when) const
Determines whether or not this date falls after the specified time.
- bool **before** (const **Date** &when) const
Determines whether or not this date falls before the specified time.
- std::string **toString** () const
*Converts this **Date** (p. 1467) object to a String of the form:.*
- virtual int **compareTo** (const **Date** &value) const
Compares this Data object to the one given.
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Date** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.265.1 Detailed Description

Wrapper class around a time value in milliseconds. This class is comparable to Java's `java.util.Date` class.

Since:

1.0

6.265.2 Constructor & Destructor Documentation

6.265.2.1 `decaf::util::Date::Date ()`

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

6.265.2.2 `decaf::util::Date::Date (long long milliseconds)`

Constructs the date with a given time value.

Parameters:

milliseconds The time in milliseconds;

6.265.2.3 `decaf::util::Date::Date (const Date & source)`

Copy constructor.

Parameters:

source The **Date** (p. 1467) instance to copy into this one.

6.265.2.4 `virtual decaf::util::Date::~~Date ()` [virtual]

6.265.3 Member Function Documentation

6.265.3.1 `bool decaf::util::Date::after (const Date & when) const`

Determines whether or not this date falls after the specified time.

Parameters:

when The date to compare

Returns:

true if this date falls after when.

6.265.3.2 `bool decaf::util::Date::before (const Date & when) const`

Determines whether or not this date falls before the specified time.

Parameters:

when The date to compare

Returns:

true if this date falls before when.

6.265.3.3 virtual int decaf::util::Date::compareTo (const Date & *value*) const
[virtual]

Compares this Date object to the one given.

Parameters:

value The **Date** (p. 1467) value to compare to this one.

Returns:

zero if the **Date** (p. 1467) values are equal, a value less than zero if this Date value is earlier than argument value, and a value greater than zero if this **Date** (p. 1467) object is later than the argument **Date** (p. 1467) value.

6.265.3.4 virtual bool decaf::util::Date::equals (const Date & *value*) const
[virtual]**Returns:**

true if this value is considered equal to the passed value.

6.265.3.5 long long decaf::util::Date::getTime () const

Gets the underlying time.

Returns:

The underlying time value in milliseconds.

6.265.3.6 virtual bool decaf::util::Date::operator< (const Date & *value*) const
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.265.3.7 Date& decaf::util::Date::operator= (const Date & *value*)

Assigns the value of one **Date** (p. 1467) object to another.

Parameters:

value The value to be copied into this **Date** (p. 1467) object.

Returns:

reference to this object with the newly assigned value.

6.265.3.8 virtual bool decaf::util::Date::operator== (const Date & *value*) const
[virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.265.3.9 void decaf::util::Date::setTime (long long *milliseconds*)

Sets the underlying time.

Parameters:

milliseconds The underlying time value in milliseconds.

6.265.3.10 std::string decaf::util::Date::toString () const

Converts this **Date** (p. 1467) object to a String of the form: . dow mon dd hh:mm:ss zzz yyyy where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.
- mm is the minute within the hour (00 through 59), as two decimal digits.
- ss is the second within the minute (00 through 61, as two decimal digits.
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.

- yyyy is the year, as four decimal digits.

Returns:

the String representation of the **Date** (p. 1467) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

6.266 decaf::internal::DecafRuntime Class Reference

Handles APR initialization and termination.

#include <src/main/decaf/internal/DecafRuntime.h> Inheritance diagram for decaf::internal::DecafRuntime:

Public Member Functions

- **DecafRuntime ()**
Initializes the APR Runtime for a library.
- virtual **~DecafRuntime ()**
Terminates the APR Runtime for a library.
- apr_pool_t * **getGlobalPool () const**
Grants access to the Global APR Pool instance that should be used when creating new threads.

6.266.1 Detailed Description

Handles APR initialization and termination.

6.266.2 Constructor & Destructor Documentation

6.266.2.1 decaf::internal::DecafRuntime::DecafRuntime ()

Initializes the APR Runtime for a library.

6.266.2.2 virtual decaf::internal::DecafRuntime::~~DecafRuntime () [virtual]

Terminates the APR Runtime for a library.

6.266.3 Member Function Documentation

6.266.3.1 apr_pool_t* decaf::internal::DecafRuntime::getGlobalPool () const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**DecafRuntime.h**

6.267 activemq::threads::DedicatedTaskRunner Class Reference

#include <src/main/activemq/threads/DedicatedTaskRunner.h> Inheritance diagram for activemq::threads::DedicatedTaskRunner:

Public Member Functions

- **DedicatedTaskRunner** (**Task** *task)
- virtual **~DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

- virtual void **shutdown** ()

Shutdown once the task has finished and the TaskRunner's thread has exited.

- virtual void **wakeup** ()

*Signal the **TaskRunner** (p. 3150) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3148) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.267.1 Constructor & Destructor Documentation

6.267.1.1 **activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner** (**Task** *task)

6.267.1.2 **virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner** () [virtual]

6.267.2 Member Function Documentation

6.267.2.1 **virtual void activemq::threads::DedicatedTaskRunner::run** () [protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2816).

6.267.2.2 virtual void activemq::threads::DedicatedTaskRunner::shutdown () [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3150).

6.267.2.3 virtual void activemq::threads::DedicatedTaskRunner::shutdown (unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters:

timeout - Time in Milliseconds to wait for the task to stop.

Implements **activemq::threads::TaskRunner** (p. 3150).

6.267.2.4 virtual void activemq::threads::DedicatedTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 3150) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3148) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3151).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**

6.268 activemq::transport::DefaultTransportListener Class Reference

#include <src/main/activemq/transport/DefaultTransportListener.h> Inheritance diagram for activemq::transport::DefaultTransportListener:

Public Member Functions

- virtual **~DefaultTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command *AMQCPP_UNUSED*)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex *AMQCPP_UNUSED*)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.

6.268.1 Constructor & Destructor Documentation

- 6.268.1.1** virtual **activemq::transport::DefaultTransportListener::~~DefaultTransportListener** () [inline, virtual]

6.268.2 Member Function Documentation

- 6.268.2.1** virtual void **activemq::transport::DefaultTransportListener::onCommand** (const **Pointer**< **Command** > &command *AMQCPP_UNUSED*) [inline, virtual]

Event handler for the receipt of a command. The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3273) deletes the command upon receipt.

Parameters:

command the received command object.

- 6.268.2.2** virtual void **activemq::transport::DefaultTransportListener::onException** (const **decaf::lang::Exception** &ex *AMQCPP_UNUSED*) [inline, virtual]

Event handler for an exception from a command transport.

Parameters:

ex The exception.

6.268.2.3 `virtual void activemq::transport::DefaultTransportListener::transportInterrupted ()`
[inline, virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3290).

6.268.2.4 `virtual void activemq::transport::DefaultTransportListener::transportResumed ()`
[inline, virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3290).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/DefaultTransportListener.h`

6.269 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

#include <src/main/decaf/util/concurrent/Delayed.h> Inheritance diagram for decaf::util::concurrent::Delayed:

Public Member Functions

- virtual `~Delayed()`
- virtual long long `getDelay` (const `TimeUnit` &unit)=0

Returns the remaining delay associated with this object, in the given time unit.

6.269.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay. An implementation of this interface must define a `Comparable` methods that provides an ordering consistent with its `getDelay` method.

6.269.2 Constructor & Destructor Documentation

6.269.2.1 virtual `decaf::util::concurrent::Delayed::~~Delayed()` [`inline`, `virtual`]

6.269.3 Member Function Documentation

6.269.3.1 virtual long long `decaf::util::concurrent::Delayed::getDelay` (const `TimeUnit` & *unit*) [`pure virtual`]

Returns the remaining delay associated with this object, in the given time unit.

Parameters:

unit The time unit

Returns:

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Delayed.h`

6.270 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

```
#include <src/main/cms/DeliveryMode.h>
```

Public Types

- enum **DELIVERY_MODE** { **PERSISTENT** = 0, **NON_PERSISTENT** = 1 }

*Enumeration values for **Message** (p. 2163) Delivery Mode.*

Public Member Functions

- virtual **~DeliveryMode** ()

6.270.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages. When a client sends a **cms::Message** (p. 2163) it can mark the **Message** (p. 2163) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 2163) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 2163) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 2163) throughput.

The **DeliveryMode** (p. 1478) covers only the transport of the **Message** (p. 2163) for sending client to its destination and doesn't apply to the receiving **Message** (p. 2163) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 2163) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 2163) consumer allows for it.

Since:

1.0

6.270.2 Member Enumeration Documentation

6.270.2.1 enum cms::DeliveryMode::DELIVERY_MODE

Enumeration values for **Message** (p. 2163) Delivery Mode.

Enumerator:

PERSISTENT

NON_PERSISTENT

6.270.3 Constructor & Destructor Documentation

6.270.3.1 virtual cms::DeliveryMode::~~DeliveryMode () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**DeliveryMode.h**

6.271 cms::Destination Class Reference

A **Destination** (p. 1480) object encapsulates a provider-specific address.

#include <src/main/cms/Destination.h> Inheritance diagram for cms::Destination:

Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY_TOPIC**, **TEMPORARY_QUEUE** }

Public Member Functions

- virtual **~Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0
*Retrieve the **Destination** (p. 1480) Type for this **Destination** (p. 1480).*
- virtual **cms::Destination * clone** () const =0
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)=0
*Copies the contents of the given **Destination** (p. 1480) object to this one.*
- virtual const **CMSProperties** & **getCMSProperties** () const =0
Retrieve any properties that might be part of the destination that was specified.

6.271.1 Detailed Description

A **Destination** (p. 1480) object encapsulates a provider-specific address. There is no standard definition of a **Destination** (p. 1480) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1480) address.

All CMS **Destination** (p. 1480) objects support concurrent use.

Since:

1.0

6.271.2 Member Enumeration Documentation

6.271.2.1 enum cms::Destination::DestinationType

Enumerator:

TOPIC
QUEUE
TEMPORARY_TOPIC
TEMPORARY_QUEUE

6.271.3 Constructor & Destructor Documentation

6.271.3.1 virtual cms::Destination::~~Destination () [inline, virtual]

6.271.4 Member Function Documentation

6.271.4.1 virtual cms::Destination* cms::Destination::clone () const [pure virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implemented in **activemq::commands::ActiveMQQueue** (p. 420),
activemq::commands::ActiveMQTempQueue (p. 524), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 549), and **ac-**
tivemq::commands::ActiveMQTopic (p. 599).

6.271.4.2 virtual void cms::Destination::copy (const cms::Destination & *source*)
[pure virtual]

Copies the contents of the given **Destination** (p. 1480) object to this one.

Parameters:

source The source **Destination** (p. 1480) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 420),
activemq::commands::ActiveMQTempQueue (p. 524), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 549), and **ac-**
tivemq::commands::ActiveMQTopic (p. 599).

6.271.4.3 virtual const CMSProperties& cms::Destination::getCMSProperties ()
const [pure virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

A {const} reference to a **CMSProperties** (p. 1036) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 421),
activemq::commands::ActiveMQTempQueue (p. 525), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 550), and **ac-**
tivemq::commands::ActiveMQTopic (p. 600).

6.271.4.4 virtual DestinationType cms::Destination::getDestinationType () const
[pure virtual]

Retrieve the **Destination** (p. 1480) Type for this **Destination** (p. 1480).

Returns:

The **Destination** (p. 1480) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 422),
activemq::commands::ActiveMQTempQueue (p. 526), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 551), and **ac-**
tivemq::commands::ActiveMQTopic (p. 601).

The documentation for this class was generated from the following file:

- `src/main/cms/`**Destination.h**

6.272 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Static Public Attributes

- static const std::string ANY_CHILD
- static const std::string ANY_DESCENDENT

6.272.1 Field Documentation

6.272.1.1 const std::string
activemq::commands::ActiveMQDestination::DestinationFilter::ANY_ -
CHILD [static]

6.272.1.2 const std::string
activemq::commands::ActiveMQDestination::DestinationFilter::ANY_ -
DESCENDENT [static]

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/ActiveMQDestination.h

6.273 activemq::commands::DestinationInfo Class Reference

#include <src/main/activemq/commands/DestinationInfo.h> Inheritance diagram for activemq::commands::DestinationInfo:

Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DestinationInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char operationType)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_DESTINATIONINFO** = 8

Protected Member Functions

- **DestinationInfo** (const **DestinationInfo** &)
- **DestinationInfo** & **operator=** (const **DestinationInfo** &)

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **ActiveMQDestination** > **destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**

6.273.1 Constructor & Destructor Documentation

6.273.1.1 **activemq::commands::DestinationInfo::DestinationInfo** (const **DestinationInfo** &) [inline, protected]

6.273.1.2 **activemq::commands::DestinationInfo::DestinationInfo** ()

6.273.1.3 **virtual** **activemq::commands::DestinationInfo::~~DestinationInfo** () [virtual]

6.273.2 Member Function Documentation

6.273.2.1 **virtual** **DestinationInfo*** **activemq::commands::DestinationInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.273.2.2 **virtual void** **activemq::commands::DestinationInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.273.2.3 `virtual bool activemq::commands::DestinationInfo::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.273.2.4 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ()` [virtual]

6.273.2.5 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () const` [virtual]

6.273.2.6 `virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ()` [virtual]

6.273.2.7 `virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () const` [virtual]

6.273.2.8 `virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.273.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ()`
[virtual]
- 6.273.2.10 `virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () const`
[virtual]
- 6.273.2.11 `virtual unsigned char activemq::commands::DestinationInfo::getOperationType ()`
const [virtual]
- 6.273.2.12 `virtual long long activemq::commands::DestinationInfo::getTimeout ()`
const [virtual]
- 6.273.2.13 `DestinationInfo& activemq::commands::DestinationInfo::operator=`
(const DestinationInfo &) [inline, protected]
- 6.273.2.14 `virtual void activemq::commands::DestinationInfo::setBrokerPath`
(const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*)
[virtual]
- 6.273.2.15 `virtual void activemq::commands::DestinationInfo::setConnectionId`
(const Pointer< ConnectionId > & *connectionId*) [virtual]
- 6.273.2.16 `virtual void activemq::commands::DestinationInfo::setDestination`
(const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.273.2.17 `virtual void activemq::commands::DestinationInfo::setOperationType`
(unsigned char *operationType*) [virtual]
- 6.273.2.18 `virtual void activemq::commands::DestinationInfo::setTimeout (long`
long *timeout*) [virtual]
- 6.273.2.19 `virtual std::string activemq::commands::DestinationInfo::toString ()`
const [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

- 6.273.2.20 `virtual Pointer<Command> activemq::commands::DestinationInfo::visit`
(activemq::state::CommandVisitor * *visitor*) throw (
exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.273.3 Field Documentation

- 6.273.3.1** `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::DestinationInfo::brokerPath` [protected]
- 6.273.3.2** `Pointer<ConnectionId>` `activemq::commands::DestinationInfo::connectionId`
[protected]
- 6.273.3.3** `Pointer<ActiveMQDestination>` `activemq::commands::DestinationInfo::destination`
[protected]
- 6.273.3.4** `const unsigned char` `activemq::commands::DestinationInfo::ID_DESTINATIONINFO = 8` [static]
- 6.273.3.5** `unsigned char` `activemq::commands::DestinationInfo::operationType`
[protected]
- 6.273.3.6** `long long` `activemq::commands::DestinationInfo::timeout` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DestinationInfo.h`

6.274 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1489).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.274.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1489). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.274.2 Constructor & Destructor Documentation

6.274.2.1 `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoM`
`() [inline]`

6.274.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~~DestinationInfoM`
`() [inline, virtual]`

6.274.3 Member Function Documentation

6.274.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.274.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.274.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 686).

6.274.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 687).

6.274.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 688).

6.274.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.274.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**

6.275 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1493).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.275.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1493). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.275.2 Constructor & Destructor Documentation

6.275.2.1 `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoM`
`() [inline]`

6.275.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~~DestinationInfoM`
`() [inline, virtual]`

6.275.3 Member Function Documentation

6.275.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.275.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.275.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.275.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.275.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.275.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.275.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**

6.276 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1497).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.276.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1497). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.276.2 Constructor & Destructor Documentation

6.276.2.1 `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::DestinationInfoM`
`() [inline]`

6.276.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::~~DestinationInfoM`
`() [inline, virtual]`

6.276.3 Member Function Documentation

6.276.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.276.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.276.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 672).

6.276.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 673).

6.276.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 674).

6.276.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.276.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DestinationInfoMarshaller.h**

6.277 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1501).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.277.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1501). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.277.2 Constructor & Destructor Documentation

6.277.2.1 `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoM`
`() [inline]`

6.277.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~~DestinationInfoM`
`() [inline, virtual]`

6.277.3 Member Function Documentation

6.277.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.277.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.277.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 665).

6.277.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 666).

6.277.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 667).

6.277.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.277.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**

6.278 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1505).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.278.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1505). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.278.2 Constructor & Destructor Documentation

6.278.2.1 `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::DestinationInfoM
() [inline]`

6.278.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::~~DestinationInfoM
() [inline, virtual]`

6.278.3 Member Function Documentation

6.278.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.278.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.278.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 679).

6.278.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 680).

6.278.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 681).

6.278.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.278.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DestinationInfoMarshaller.h**

6.279 activemq::cmsutil::DestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

#include <src/main/activemq/cmsutil/DestinationResolver.h> Inheritance diagram for `activemq::cmsutil::DestinationResolver`:

Public Member Functions

- virtual `~DestinationResolver()`
- virtual void `init(ResourceLifecycleManager *mgr)=0`
Initializes this destination resolver for use.
- virtual void `destroy()`=0
Destroys any allocated resources.
- virtual `cms::Destination * resolveDestinationName(cms::Session *session, const std::string &destName, bool pubSubDomain)=0` throw (`cms::CMSException`)
Resolves the given name to a destination.

6.279.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.279.2 Constructor & Destructor Documentation

- 6.279.2.1** virtual `activemq::cmsutil::DestinationResolver::~~DestinationResolver()` [inline, virtual]

6.279.3 Member Function Documentation

- 6.279.3.1** virtual void `activemq::cmsutil::DestinationResolver::destroy()` [pure virtual]

Destroys any allocated resources.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1568).

- 6.279.3.2** virtual void `activemq::cmsutil::DestinationResolver::init(ResourceLifecycleManager *mgr)` [pure virtual]

Initializes this destination resolver for use. Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1509)).

Parameters:

mgr the resource lifecycle manager.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1569).

6.279.3.3 `virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain) throw (cms::CMSException)` [pure virtual]

Resolves the given name to a destination. If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters:

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns:

the resolved destination

Exceptions:

cms::CMSException (p. 1031) if resolution failed.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1569).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DestinationResolver.h`

6.280 activemq::commands::DiscoveryEvent Class Reference

#include <src/main/activemq/commands/DiscoveryEvent.h> Inheritance diagram for activemq::commands::DiscoveryEvent:

Public Member Functions

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DiscoveryEvent** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &serviceName)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)

Static Public Attributes

- static const unsigned char **ID_DISCOVERYEVENT** = 40

Protected Member Functions

- **DiscoveryEvent** (const **DiscoveryEvent** &)
- **DiscoveryEvent** & **operator=** (const **DiscoveryEvent** &)

Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

6.280.1 Constructor & Destructor Documentation

6.280.1.1 `activemq::commands::DiscoveryEvent::DiscoveryEvent (const DiscoveryEvent &) [inline, protected]`

6.280.1.2 `activemq::commands::DiscoveryEvent::DiscoveryEvent ()`

6.280.1.3 `virtual activemq::commands::DiscoveryEvent::~~DiscoveryEvent () [virtual]`

6.280.2 Member Function Documentation

6.280.2.1 `virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.280.2.2 `virtual void activemq::commands::DiscoveryEvent::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

6.280.2.3 `virtual bool activemq::commands::DiscoveryEvent::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

- 6.280.2.4 `virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName ()`
[virtual]
- 6.280.2.5 `virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName ()`
`const` [virtual]
- 6.280.2.6 `virtual unsigned char activemq::commands::DiscoveryEvent::getDataSetType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

- 6.280.2.7 `virtual std::string& activemq::commands::DiscoveryEvent::getServiceName ()`
[virtual]
- 6.280.2.8 `virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName ()`
`const` [virtual]
- 6.280.2.9 `DataSet& activemq::commands::DiscoveryEvent::operator=`
`(const DataSet &)` [inline, protected]
- 6.280.2.10 `virtual void activemq::commands::DiscoveryEvent::setBrokerName`
`(const std::string & brokerName)` [virtual]
- 6.280.2.11 `virtual void activemq::commands::DiscoveryEvent::setServiceName`
`(const std::string & serviceName)` [virtual]
- 6.280.2.12 `virtual std::string activemq::commands::DiscoveryEvent::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 718).

6.280.3 Field Documentation

- 6.280.3.1 `std::string activemq::commands::DiscoveryEvent::brokerName`
[protected]
- 6.280.3.2 `const unsigned char activemq::commands::DiscoveryEvent::ID_ -`
`DISCOVERYEVENT = 40` [static]
- 6.280.3.3 `std::string activemq::commands::DiscoveryEvent::serviceName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DiscoveryEvent.h`

6.281 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1515).

#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.281.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1515). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.281.2 Constructor & Destructor Documentation

6.281.2.1 `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.281.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.281.3 Member Function Documentation

6.281.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.281.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.281.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.281.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.281.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.281.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.281.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h

6.282 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1519).

#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.282.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1519). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.282.2 Constructor & Destructor Documentation

6.282.2.1 `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.282.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.282.3 Member Function Documentation

6.282.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.282.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.282.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.282.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.282.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.282.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.282.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h

6.283 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1523).

#include <src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.283.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1523). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.283.2 Constructor & Destructor Documentation

6.283.2.1 `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::DiscoveryEventM`
`() [inline]`

6.283.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::~~DiscoveryEvent`
`() [inline, virtual]`

6.283.3 Member Function Documentation

6.283.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.283.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.283.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.283.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.283.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.283.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.283.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h

6.284 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1527).

#include <src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.284.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1527). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.284.2 Constructor & Destructor Documentation

6.284.2.1 `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::DiscoveryEventM
() [inline]`

6.284.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::~~DiscoveryEvent
() [inline, virtual]`

6.284.3 Member Function Documentation

6.284.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.284.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.284.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.284.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.284.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.284.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.284.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

6.285 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1531).

#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.285.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1531). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.285.2 Constructor & Destructor Documentation

6.285.2.1 `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventM
() [inline]`

6.285.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~~DiscoveryEvent
() [inline, virtual]`

6.285.3 Member Function Documentation

6.285.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.285.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.285.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.285.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.285.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.285.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.285.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h

6.286 activemq::core::DispatchData Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

Public Member Functions

- **DispatchData** ()
- **DispatchData** (const decaf::lang::Pointer< commands::ConsumerId > &consumer, const decaf::lang::Pointer< commands::Message > &message)
- const decaf::lang::Pointer< commands::ConsumerId > & **getConsumerId** ()
- const decaf::lang::Pointer< commands::Message > & **getMessage** ()

6.286.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

6.286.2 Constructor & Destructor Documentation

6.286.2.1 `activemq::core::DispatchData::DispatchData ()` [inline]

6.286.2.2 `activemq::core::DispatchData::DispatchData (const decaf::lang::Pointer< commands::ConsumerId > & consumer, const decaf::lang::Pointer< commands::Message > & message)` [inline]

6.286.3 Member Function Documentation

6.286.3.1 `const decaf::lang::Pointer<commands::ConsumerId>& activemq::core::DispatchData::getConsumerId ()` [inline]

6.286.3.2 `const decaf::lang::Pointer<commands::Message>& activemq::core::DispatchData::getMessage ()` [inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/DispatchData.h`

6.287 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

#include <src/main/activemq/core/Dispatcher.h> Inheritance diagram for activemq::core::Dispatcher:

Public Member Functions

- virtual **~Dispatcher** ()
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)=0
Dispatches a message to a particular consumer.

6.287.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

6.287.2 Constructor & Destructor Documentation

6.287.2.1 virtual **activemq::core::Dispatcher::~Dispatcher** () [inline, virtual]

6.287.3 Member Function Documentation

6.287.3.1 virtual void **activemq::core::Dispatcher::dispatch** (const **Pointer**< **MessageDispatch** > & *message*) [pure virtual]

Dispatches a message to a particular consumer.

Parameters:

message - the message to be dispatched.

Implemented in **activemq::core::ActiveMQConsumer** (p. 267), and **activemq::core::ActiveMQSession** (p. 453).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Dispatcher.h**

6.288 decaf::lang::Double Class Reference

#include <src/main/decaf/lang/Double.h> Inheritance diagram for decaf::lang::Double:

Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Double** ()
- virtual int **compareTo** (const **Double** &d) const
*Compares this **Double** (p. 1537) instance with another.*
- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Double** &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const double &d) const
*Compares this **Double** (p. 1537) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const double &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const

Answers the long value which the receiver represents.

- `bool isInfinite () const`
- `bool isNaN () const`

Static Public Member Functions

- `static int compare (double d1, double d2)`
Compares the two specified double values.
- `static long long doubleToLongBits (double value)`
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.
- `static long long doubleToRawLongBits (double value)`
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.
- `static bool isInfinite (double value)`
- `static bool isNaN (double value)`
- `static double longBitsToDouble (long long bits)`
Returns the double value corresponding to a given bit representation.
- `static double parseDouble (const std::string &value) throw (exceptions::NumberFormatException)`
*Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p. 1537).*
- `static std::string toHexString (double value)`
Returns a hexadecimal string representation of the double argument.
- `static std::string toString (double value)`
Returns a string representation of the double argument.
- `static Double valueOf (double value)`
*Returns a **Double** (p. 1537) instance representing the specified double value.*
- `static Double valueOf (const std::string &value) throw (exceptions::NumberFormatException)`
*Returns a **Double** (p. 1537) instance that wraps a primitive double which is parsed from the string value passed.*

Static Public Attributes

- `static const int SIZE = 64`
The size in bits of the primitive int type.
- `static const double MAX_VALUE`
The maximum value that the primitive type can hold.

- static const double **MIN_VALUE**
The minimum value that the primitive type can hold.
- static const double **NaN**
*Constant for the Not a **Number** (p. 2432) Value.*
- static const double **POSITIVE_INFINITY**
Constant for Positive Infinity.
- static const double **NEGATIVE_INFINITY**
Constant for Negative Infinity.

6.288.1 Constructor & Destructor Documentation

6.288.1.1 decaf::lang::Double::Double (double *value*)

Parameters:

value - the primitive type to wrap

6.288.1.2 decaf::lang::Double::Double (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - the string to convert to a primitive type to wrap

6.288.1.3 virtual decaf::lang::Double::~~Double () [inline, virtual]

6.288.2 Member Function Documentation

6.288.2.1 virtual unsigned char decaf::lang::Double::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2432).

6.288.2.2 static int decaf::lang::Double::compare (double *d1*, double *d2*) [static]

Compares the two specified double values. The sign of the integer value returned is the same as that of the integer that would be returned by the call: new Double(d1).compareTo(new Double(d2))

Parameters:

d1 - the first double to compare
d2 - the second double to compare

Returns:

the value 0 if d1 is numerically equal to d2; a value less than 0 if d1 is numerically less than d2; and a value greater than 0 if d1 is numerically greater than d2.

6.288.2.3 virtual int decaf::lang::Double::compareTo (const double & d) const
[virtual]

Compares this **Double** (p. 1537) instance with another.

Parameters:

d - the **Double** (p. 1537) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< double >** (p. 1083).

6.288.2.4 virtual int decaf::lang::Double::compareTo (const Double & d) const
[virtual]

Compares this **Double** (p. 1537) instance with another.

Parameters:

d - the **Double** (p. 1537) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< Double >** (p. 1083).

6.288.2.5 static long long decaf::lang::Double::doubleToLongBits (double value)
[static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout. Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long) method, will produce a floating-point value the same as the argument to doubleToLongBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters:

value - double to be converted

Returns:

the long long bits that make up the double

6.288.2.6 static long long decaf::lang::Double::doubleToRawLongBits (double *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values. Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the doubleToLongBits method, doubleToRawLongBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long) method, will produce a floating-point value the same as the argument to doubleToRawLongBits.

Parameters:

value - double to be converted

Returns:

the long long bits that make up the double

6.288.2.7 virtual double decaf::lang::Double::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.288.2.8 `bool decaf::lang::Double::equals (const double & d) const` [inline, virtual]

Parameters:

d - the **Double** (p. 1537) object to compare against.

Returns:

true if the two **Double** (p. 1537) Objects have the same value.

Implements **decaf::lang::Comparable**< **double** > (p. 1084).

6.288.2.9 `bool decaf::lang::Double::equals (const Double & d) const` [inline, virtual]

Parameters:

d - the **Double** (p. 1537) object to compare against.

Returns:

true if the two **Double** (p. 1537) Objects have the same value.

Implements **decaf::lang::Comparable**< **Double** > (p. 1084).

6.288.2.10 `virtual float decaf::lang::Double::floatValue () const` [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.288.2.11 `virtual int decaf::lang::Double::intValue () const` [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.288.2.12 `static bool decaf::lang::Double::isInfinite (double value)` [static]

Parameters:

value - The double to check.

Returns:

true if the double is equal to infinity.

6.288.2.13 `bool decaf::lang::Double::isInfinite () const`**Returns:**

true if the double is equal to positive infinity.

6.288.2.14 `static bool decaf::lang::Double::isNaN (double value) [static]`**Parameters:**

value - The double to check.

Returns:

true if the double is equal to NaN.

6.288.2.15 `bool decaf::lang::Double::isNaN () const`**Returns:**

true if the double is equal to NaN.

6.288.2.16 `static double decaf::lang::Double::longBitsToDouble (long long bits) [static]`

Returns the double value corresponding to a given bit representation. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1541) method.

Parameters:

bits - the long long bits to convert to double

Returns:

the double converted from the bits

6.288.2.17 `virtual long long decaf::lang::Double::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.288.2.18 `virtual bool decaf::lang::Double::operator< (const double & d) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1084).

6.288.2.19 `virtual bool decaf::lang::Double::operator< (const Double & d) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< Double >` (p.1084).

6.288.2.20 `virtual bool decaf::lang::Double::operator== (const double & d) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1085).

6.288.2.21 `virtual bool decaf::lang::Double::operator== (const Double & d) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Double** > (p. 1085).

6.288.2.22 **static double decaf::lang::Double::parseDouble (const std::string *value*)
 throw (exceptions::NumberFormatException)** [static]

Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p. 1537).

Parameters:

value - The string to parse to an double

Returns:

a double parsed from the passed string

Exceptions:

NumberFormatException

6.288.2.23 **virtual short decaf::lang::Double::shortValue () const** [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2434).

6.288.2.24 **static std::string decaf::lang::Double::toHexString (double *value*)
 [static]**

Returns a hexadecimal string representation of the double argument. All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 1774) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed

by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters:

value - The double to convert to a string

Returns:

the Hex formatted double string.

6.288.2.25 static std::string decaf::lang::Double::toString (double *value*) [static]

Returns a string representation of the double argument. All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m:

- o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".
- o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".
- o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m.
- o If m is less than 10⁻³ or greater than or equal to 10⁷, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10ⁿ ≤ m < 10ⁿ⁺¹; then let a be the mathematically exact quotient of m and 10ⁿ so that 1 ≤ a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1774).

Parameters:

value - The double to convert to a string

Returns:

the formatted double string.

6.288.2.26 std::string decaf::lang::Double::toString () const

Returns:

this **Double** (p. 1537) Object as a String Representation

6.288.2.27 static Double decaf::lang::Double::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Double** (p. 1537) instance that wraps a primitive double which is parsed from the string value passed.

Parameters:

value - the string to parse

Returns:

a new **Double** (p. 1537) instance wrapping the double parsed from value

Exceptions:

NumberFormatException on error.

6.288.2.28 static Double decaf::lang::Double::valueOf (double value) [static]

Returns a **Double** (p. 1537) instance representing the specified double value.

Parameters:

value - double to wrap

Returns:

new **Double** (p. 1537) instance wrapping the primitive value

6.288.3 Field Documentation**6.288.3.1 const double decaf::lang::Double::MAX_VALUE [static]**

The maximum value that the primitive type can hold.

6.288.3.2 const double decaf::lang::Double::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.288.3.3 const double decaf::lang::Double::NaN [static]

Constant for the Not a **Number** (p. 2432) Value.

6.288.3.4 const double decaf::lang::Double::NEGATIVE_INFINITY [static]

Constant for Negative Infinity.

6.288.3.5 const double decaf::lang::Double::POSITIVE_INFINITY [static]

Constant for Positive Infinity.

6.288.3.6 const int decaf::lang::Double::SIZE = 64 [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Double.h**

6.289 decaf::internal::nio::DoubleArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h> Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

Public Member Functions

- **DoubleArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **DoubleArrayBuffer** (p. 1548) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **DoubleArrayBuffer** (double *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **DoubleArrayBuffer** (p. 1548) object that wraps the given array.*
- **DoubleArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t length, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset.*
- **DoubleArrayBuffer** (const DoubleArrayBuffer &other)
*Create a **DoubleArrayBuffer** (p. 1548) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~**DoubleArrayBuffer** ()
- virtual double * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the double array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual DoubleBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only double buffer that shares this buffer's content.
- virtual DoubleBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual DoubleBuffer * **duplicate** ()
Creates a new double buffer that shares this buffer's content.
- virtual double **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.

- virtual double **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible double array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual DoubleBuffer & **put** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual DoubleBuffer & **put** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual DoubleBuffer * **slice** () const
Creates a new DoubleBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteArrayListBuffer** (p. 870) as Read-Only.*

6.289.1 Constructor & Destructor Documentation

6.289.1.1 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **DoubleArrayBuffer** (p. 1548) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.289.1.2 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **DoubleArrayBuffer** (p. 1548) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap
offset - the position that is this buffers start pos.
capacity - size of the array, this is the limit we read and write to.
readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.289.1.3 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t length, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset. The capacity and limit of the new **DoubleArrayBuffer** (p. 1548) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 908) to wrap
offset - the offset into array where the buffer starts
length - the length of the array we are wrapping or limit.
readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.289.1.4 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const DoubleArrayBuffer & other)`

Create a **DoubleArrayBuffer** (p. 1548) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.

Parameters:

other - the **DoubleArrayBuffer** (p. 1548) this one is to mirror.

6.289.1.5 `virtual decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer () [virtual]`

6.289.2 Member Function Documentation

6.289.2.1 `virtual double* decaf::internal::nio::DoubleArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the double array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1559).

6.289.2.2 `virtual std::size_t decaf::internal::nio::DoubleArrayBuffer::arrayOffset
() throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1559).

6.289.2.3 `virtual DoubleBuffer* de-
caf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer ()
const [virtual]`

Creates a new, read-only double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1559).

6.289.2.4 **virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact** () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this DoubleBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1560).

6.289.2.5 **virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::duplicate** () [virtual]

Creates a new double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new double Buffer which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1560).

6.289.2.6 **virtual double decaf::internal::nio::DoubleArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)** [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the double is to be read

Returns:

the double that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::DoubleBuffer** (p. 1562).

6.289.2.7 virtual double decaf::internal::nio::DoubleArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the double at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::DoubleBuffer** (p. 1562).

6.289.2.8 virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible double array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1562).

6.289.2.9 virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 806).

6.289.2.10 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1563).

6.289.2.11 **virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (double *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)** [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1563).

6.289.2.12 **virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly (bool *value*)** [inline, protected, virtual]

Sets this **ByteBuffer** (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.289.2.13 **virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice () const** [virtual]

Creates a new **DoubleBuffer** whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **DoubleBuffer** which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1565).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/DoubleArrayBuffer.h`

6.290 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:

#include <src/main/decaf/nio/DoubleBuffer.h> Inheritance diagram for decaf::nio::DoubleBuffer:

Public Member Functions

- virtual **~DoubleBuffer** ()
- virtual std::string **toString** () const
- virtual double * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the double array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **DoubleBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only double buffer that shares this buffer's content.
- virtual **DoubleBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **DoubleBuffer** * **duplicate** ()=0
Creates a new double buffer that shares this buffer's content.
- virtual double **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual double **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **DoubleBuffer** & **get** (std::vector< double > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **DoubleBuffer** & **get** (double *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible double array.
- **DoubleBuffer** & **put** (**DoubleBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the doubles remaining in the given source buffer into this buffer.

- **DoubleBuffer** & **put** (const double *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers doubles into this buffer from the given source array.

- **DoubleBuffer** & **put** (std::vector< double > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source doubles array into this buffer.

- virtual **DoubleBuffer** & **put** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual **DoubleBuffer** & **put** (std::size_t index, double value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes the given doubles into this buffer at the given index.

- virtual **DoubleBuffer** * **slice** () const =0

*Creates a new **DoubleBuffer** (p. 1556) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **DoubleBuffer** &value) const

Compares this object with the specified object for order.

- virtual bool **equals** (const **DoubleBuffer** &value) const

- virtual bool **operator==** (const **DoubleBuffer** &value) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const **DoubleBuffer** &value) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **DoubleBuffer** * **allocate** (std::size_t capacity)

Allocates a new Double buffer.

- static **DoubleBuffer** * **wrap** (double *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

*Wraps the passed buffer with a new **DoubleBuffer** (p. 1556).*

- static **DoubleBuffer** * **wrap** (std::vector< double > &buffer)

*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1556).*

Protected Member Functions

- **DoubleBuffer** (std::size_t capacity)

*Creates a **DoubleBuffer** (p. 1556) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.290.1 Detailed Description

This class defines four categories of operations upon double buffers: o Absolute and relative get and put methods that read and write single doubles; o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.290.2 Constructor & Destructor Documentation

6.290.2.1 decaf::nio::DoubleBuffer::DoubleBuffer (std::size_t capacity) [protected]

Creates a **DoubleBuffer** (p. 1556) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 803) in doubles

6.290.2.2 virtual decaf::nio::DoubleBuffer::~~DoubleBuffer () [inline, virtual]

6.290.3 Member Function Documentation

6.290.3.1 static DoubleBuffer* decaf::nio::DoubleBuffer::allocate (std::size_t capacity) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in doubles

Returns:

the **DoubleBuffer** (p. 1556) that was allocated, caller owns.

6.290.3.2 virtual double* decaf::nio::DoubleBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the double array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 803)

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1550).

6.290.3.3 virtual std::size_t decaf::nio::DoubleBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1551).

6.290.3.4 virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1551).

6.290.3.5 virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **DoubleBuffer** (p. 1556)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1552).

6.290.3.6 virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer & value) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.290.3.7 virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate () [pure virtual]

Creates a new double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new double **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1552).

6.290.3.8 virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.290.3.9 DoubleBuffer& decaf::nio::DoubleBuffer::get (double * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies `length` doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` doubles remaining in this buffer

NullPointerException if the passed buffer is null.

6.290.3.10 DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` doubles remaining in this buffer

6.290.3.11 `virtual double decaf::nio::DoubleBuffer::get (std::size_t index) const
throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 803) where the double is to be read

Returns:

the double that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1552).

6.290.3.12 `virtual double decaf::nio::DoubleBuffer::get () throw (
BufferUnderflowException) [pure virtual]`

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the double at the current position

Exceptions:

BufferUnderflowException (p. 837) if there no more data to return

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1553).

6.290.3.13 `virtual bool decaf::nio::DoubleBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible double array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1553).

6.290.3.14 `virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer &
value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.290.3.15 `virtual bool decaf::nio::DoubleBuffer::operator==(const DoubleBuffer & value) const` [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.290.3.16 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put(std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1553).

6.290.3.17 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put(double value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1554).

6.290.3.18 **DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)**

This method transfers the entire content of the given source doubles array into this buffer. This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters:

buffer - The buffer whose contents are copied to this **DoubleBuffer** (p. 1556)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.290.3.19 **DoubleBuffer& decaf::nio::DoubleBuffer::put (const double * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)**

This method transfers doubles into this buffer from the given source array. If there are more doubles to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 808), then no doubles are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters:

buffer- The array from which doubles are to be read

offset- The offset within the array of the first double to be read;

length - The number of doubles to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only
NullPointerException if the passed buffer is null.

6.290.3.20 DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the doubles remaining in the given source buffer into this buffer. If there are more doubles remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 808), then no doubles are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `n = src.remaining()` doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters:

src - the buffer to take doubles from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining doubles in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.290.3.21 virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice () const [pure virtual]

Creates a new **DoubleBuffer** (p. 1556) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **DoubleBuffer** (p. 1556) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1554).

6.290.3.22 virtual std::string decaf::nio::DoubleBuffer::toString () const [virtual]

Returns:

a `std::string` describing this object

6.290.3.23 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector<double > & buffer) [static]`

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1556). The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new **DoubleBuffer** (p. 1556) that is backed by *buffer*, caller owns.

6.290.3.24 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (double * array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **DoubleBuffer** (p. 1556). The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **DoubleBuffer** (p. 1556) that is backed by *buffer*, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/DoubleBuffer.h`

6.291 decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.292 activemq::cmsutil::DynamicDestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h> Inheritance diagram for `activemq::cmsutil::DynamicDestinationResolver`:

Data Structures

- class `SessionResolver`

Manages maps of names to topics and queues for a single session.

Public Member Functions

- virtual `~DynamicDestinationResolver ()`
- virtual void `init (ResourceLifecycleManager *mgr)`
Initializes this destination resolver for use.
- virtual void `destroy ()`
Destroys any allocated resources.
- virtual `cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName, bool pubSubDomain) throw (cms::CMSException)`
Resolves the given name to a destination.

6.292.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.292.2 Constructor & Destructor Documentation

- 6.292.2.1** virtual
`activemq::cmsutil::DynamicDestinationResolver::~~DynamicDestinationResolver ()` [virtual]

6.292.3 Member Function Documentation

- 6.292.3.1** virtual void `activemq::cmsutil::DynamicDestinationResolver::destroy ()` [virtual]

Destroys any allocated resources.

Implements `activemq::cmsutil::DestinationResolver` (p. 1509).

6.292.3.2 virtual void activemq::cmsutil::DynamicDestinationResolver::init (ResourceLifecycleManager * *mgr*) [inline, virtual]

Initializes this destination resolver for use. Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p.1568)).

Parameters:

mgr the resource lifecycle manager.

Implements **activemq::cmsutil::DestinationResolver** (p.1509).

6.292.3.3 virtual cms::Destination* activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName (cms::Session * *session*, const std::string & *destName*, bool *pubSubDomain*) throw (cms::CMSException) [virtual]

Resolves the given name to a destination. If *pubSubDomain* is true, a topic will be returned, otherwise a queue will be returned.

Parameters:

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns:

the resolved destination

Exceptions:

cms::CMSException (p. 1031) if resolution failed.

Implements **activemq::cmsutil::DestinationResolver** (p.1510).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/DynamicDestinationResolver.h

6.293 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

```
#include <src/main/decaf/util/Map.h>
```

Public Member Functions

- **Entry** ()
- virtual **~Entry** ()
- virtual const K & **getKey** () const =0
- virtual const V & **getValue** () const =0
- virtual void **setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>  
class decaf::util::Map< K, V, COMPARATOR >::Entry
```

6.293.1 Constructor & Destructor Documentation

6.293.1.1 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry
() [inline]`

6.293.1.2 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR
>::Entry::~Entry () [inline, virtual]`

6.293.2 Member Function Documentation

6.293.2.1 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR
>::Entry::getKey () const [pure virtual]`

6.293.2.2 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR
>::Entry::getValue () const [pure virtual]`

6.293.2.3 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR
>::Entry::setValue (const V & value) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

6.294 decaf::io::EOFException Class Reference

#include <src/main/decaf/io/EOFException.h> Inheritance diagram for decaf::io::EOFException:

Public Member Functions

- **EOFException** () throw ()
Default Constructor.
- **EOFException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **EOFException** (const EOFException &ex) throw ()
Copy Constructor.
- **EOFException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **EOFException** (const std::exception *cause) throw ()
Constructor.
- **EOFException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **EOFException** * clone () const
Clones this exception.
- virtual ~**EOFException** () throw ()

6.294.1 Constructor & Destructor Documentation

6.294.1.1 decaf::io::EOFException::EOFException () throw () [inline]

Default Constructor.

6.294.1.2 decaf::io::EOFException::EOFException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.294.1.3 `decaf::io::EOFException::EOFException (const EOFException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.294.1.4 `decaf::io::EOFException::EOFException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.294.1.5 `decaf::io::EOFException::EOFException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.294.1.6 `decaf::io::EOFException::EOFException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.294.1.7 `virtual decaf::io::EOFException::~~EOFException () throw () [inline, virtual]`

6.294.2 Member Function Documentation

6.294.2.1 `virtual EOFException* decaf::io::EOFException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/EOFException.h`

6.295 decaf::lang::Exception Class Reference

#include <src/main/decaf/lang/Exception.h> Inheritance diagram for decaf::lang::Exception:

Public Member Functions

- **Exception** () throw ()
Default Constructor.
- **Exception** (const **Exception** &ex) throw ()
Copy Constructor.
- **Exception** (const std::exception ***cause**) throw ()
Constructor.
- **Exception** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **Exception** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**Exception** () throw ()
- virtual std::string **getMessage** () const
Gets the message for this exception.
- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception ***cause**)
Initializes the contained cause exception with the one given.
- virtual const char * **what** () const throw ()
Implement method from std::exception.
- virtual void **setMessage** (const char *msg,...)
Sets the cause for this exception.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual **Exception** * **clone** () const
Clones this exception.

- virtual `std::vector< std::pair< std::string, int > > getStackTrace () const`
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void `printStackTrace () const`
Prints the stack trace to `std::err`.
- virtual void `printStackTrace (std::ostream &stream) const`
Prints the stack trace to the given output stream.
- virtual `std::string getStackTraceString () const`
Gets the stack trace as one contiguous string.
- virtual `Exception & operator= (const Exception &ex)`
Assignment operator.

Protected Member Functions

- virtual void `setStackTrace (const std::vector< std::pair< std::string, int > > &trace)`
- virtual void `buildMessage (const char *format, va_list &args)`

Protected Attributes

- `std::string message`
The cause of this exception.
- `std::exception * cause`
*The **Exception** (p. 1574) that caused this one to be thrown.*
- `std::vector< std::pair< std::string, int > > stackTrace`
The stack trace.

6.295.1 Constructor & Destructor Documentation

6.295.1.1 decaf::lang::Exception::Exception () throw ()

Default Constructor.

Referenced by `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException()`,
`decaf::util::concurrent::CancellationException::CancellationException()`,
`decaf::util::concurrent::ExecutionException::ExecutionException()`,
`decaf::net::HttpRetryException::HttpRetryException()`, `decaf::net::MalformedURLException::MalformedURLException()`,
`decaf::net::ProtocolException::ProtocolException()`, `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException()`,
`decaf::net::SocketTimeoutException::SocketTimeoutException()`,
`decaf::util::concurrent::TimeoutException::TimeoutException()`,
`decaf::net::UnknownHostException::UnknownHostException()`, and `decaf::net::UnknownServiceException::UnknownServiceException()`.

6.295.1.2 decaf::lang::Exception::Exception (const Exception & *ex*) throw ()

Copy Constructor.

Parameters:

ex The Exception (p.1574) instance to copy.

6.295.1.3 decaf::lang::Exception::Exception (const std::exception * *cause*) throw ()

Constructor.

Parameters:

cause **Pointer** (p.2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.295.1.4 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.295.1.5 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.295.1.6 virtual decaf::lang::Exception::~~Exception () throw () [virtual]

6.295.2 Member Function Documentation

6.295.2.1 virtual void decaf::lang::Exception::buildMessage (const char * *format*, va_list & *vargs*) [protected, virtual]

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.295.2.2 virtual Exception* decaf::lang::Exception::clone () const [virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

Copy of this **Exception** (p. 1574) object

Implements **decaf::lang::Throwable** (p. 3182).

Reimplemented in **activemq::exceptions::ActiveMQException** (p. 306), **activemq::exceptions::BrokerException** (p. 749), **decaf::io::EOFException** (p. 1573), **decaf::io::InterruptedIOException** (p. 1807), **decaf::io::IOException** (p. 1822), **decaf::io::UnsupportedEncodingException** (p. 3302), **decaf::io::UTFDataFormatException** (p. 3355), **decaf::lang::exceptions::ClassCastException** (p. 1018), **decaf::lang::exceptions::IllegalArgumentException** (p. 1722), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1725), **decaf::lang::exceptions::IllegalStateException** (p. 1728), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1732), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1739), **decaf::lang::exceptions::InterruptedException** (p. 1804), **decaf::lang::exceptions::InvalidStateException** (p. 1819), **decaf::lang::exceptions::NoSuchElementException** (p. 2425), **decaf::lang::exceptions::NullPointerException** (p. 2431), **decaf::lang::exceptions::NumberFormatException** (p. 2437), **decaf::lang::exceptions::RuntimeException** (p. 2821), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3305), **decaf::net::BindException** (p. 723), **decaf::net::ConnectException** (p. 1130), **decaf::net::HttpRetryException** (p. 1719), **decaf::net::MalformedURLException** (p. 2093), **decaf::net::NoRouteToHostException** (p. 2419), **decaf::net::PortUnreachableException** (p. 2524), **decaf::net::ProtocolException** (p. 2669), **decaf::net::SocketException** (p. 2973), **decaf::net::SocketTimeoutException** (p. 2992), **decaf::net::UnknownHostException** (p. 3296), **decaf::net::UnknownServiceException** (p. 3299), **decaf::net::URISyntaxException** (p. 3337), **decaf::nio::BufferOverflowException** (p. 836), **decaf::nio::BufferUnderflowException** (p. 839), **decaf::nio::InvalidMarkException** (p. 1815), **decaf::nio::ReadOnlyBufferException** (p. 2687), **decaf::util::concurrent::BrokenBarrierException** (p. 744), **decaf::util::concurrent::CancellationException** (p. 967), **decaf::util::concurrent::ExecutionException** (p. 1607), **decaf::util::concurrent::RejectedExecutionException** (p. 2701), and **decaf::util::concurrent::TimeoutException** (p. 3187).

6.295.2.3 `virtual const std::exception* decaf::lang::Exception::getCause () const`
[inline, virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 3183).

6.295.2.4 `virtual std::string decaf::lang::Exception::getMessage () const` [inline, virtual]

Gets the message for this exception.

Returns:

Text formatted error message

Implements **decaf::lang::Throwable** (p. 3183).

6.295.2.5 `virtual std::vector< std::pair< std::string, int> > decaf::lang::Exception::getStackTrace () const` [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown. The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

Returns:

the stack trace.

Implements **decaf::lang::Throwable** (p. 3183).

6.295.2.6 `virtual std::string decaf::lang::Exception::getStackTraceString () const`
[virtual]

Gets the stack trace as one contiguous string.

Returns:

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 3183).

6.295.2.7 `virtual void decaf::lang::Exception::initCause (const std::exception * cause)` [virtual]

Initializes the contained cause exception with the one given. A copy is made to avoid ownership issues.

Parameters:

cause The exception that was the cause of this one.

Implements **decaf::lang::Throwable** (p. 3183).

6.295.2.8 virtual Exception& decaf::lang::Exception::operator= (const Exception & *ex*) [virtual]

Assignment operator.

Parameters:

ex const reference to another **Exception** (p. 1574)

6.295.2.9 virtual void decaf::lang::Exception::printStackTrace (std::ostream & *stream*) const [virtual]

Prints the stack trace to the given output stream.

Parameters:

stream the target output stream.

Implements **decaf::lang::Throwable** (p. 3184).

6.295.2.10 virtual void decaf::lang::Exception::printStackTrace () const [virtual]

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 3184).

6.295.2.11 virtual void decaf::lang::Exception::setMark (const char * *file*, const int *lineNumber*) [virtual]

Adds a file/line number to the stack trace.

Parameters:

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

Implements **decaf::lang::Throwable** (p. 3184).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`, and `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

6.295.2.12 virtual void decaf::lang::Exception::setMessage (const char * *msg*, ...) [virtual]

Sets the cause for this exception.

Parameters:

msg the format string for the msg.
... params to format into the string

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

6.295.2.13 `virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & trace) [protected, virtual]`

6.295.2.14 `virtual const char* decaf::lang::Exception::what () const throw () [inline, virtual]`

Implement method from `std::exception`.

Returns:

the const char* of `getMessage()` (p. 1578).

6.295.3 Field Documentation

6.295.3.1 `std::exception* decaf::lang::Exception::cause [protected]`

The **Exception** (p. 1574) that caused this one to be thrown.

6.295.3.2 `std::string decaf::lang::Exception::message [protected]`

The cause of this exception.

6.295.3.3 `std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace [protected]`

The stack trace.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Exception.h`

6.296 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1581) that is registered with the `Connection` (p. 1131).

```
#include <src/main/cms/ExceptionListener.h>
```

Public Member Functions

- virtual `~ExceptionListener ()`
- virtual void `onException (const cms::CMSException &ex)=0`
Called when an exception occurs.

6.296.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1581) that is registered with the `Connection` (p. 1131). An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since:

1.0

6.296.2 Constructor & Destructor Documentation

6.296.2.1 virtual `cms::ExceptionListener::~ExceptionListener ()` [inline, virtual]

6.296.3 Member Function Documentation

6.296.3.1 virtual void `cms::ExceptionListener::onException (const cms::CMSException & ex)` [pure virtual]

Called when an exception occurs. Once notified of an exception the caller should no longer use the resource that generated the exception.

Parameters:

ex Exception Object that occurred.

The documentation for this class was generated from the following file:

- `src/main/cms/ExceptionListener.h`

6.297 activemq::commands::ExceptionResponse Class Reference

#include <src/main/activemq/commands/ExceptionResponse.h> Inheritance diagram for activemq::commands::ExceptionResponse:

Public Member Functions

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ExceptionResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

Static Public Attributes

- static const unsigned char **ID_EXCEPTIONRESPONSE** = 31

Protected Member Functions

- **ExceptionResponse** (const **ExceptionResponse** &)
- **ExceptionResponse** & **operator=** (const **ExceptionResponse** &)

Protected Attributes

- **Pointer**< **BrokerError** > **exception**

6.297.1 Constructor & Destructor Documentation

6.297.1.1 `activemq::commands::ExceptionResponse::ExceptionResponse (const ExceptionResponse &) [inline, protected]`

6.297.1.2 `activemq::commands::ExceptionResponse::ExceptionResponse ()`

6.297.1.3 `virtual activemq::commands::ExceptionResponse::~~ExceptionResponse () [virtual]`

6.297.2 Member Function Documentation

6.297.2.1 `virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 2782).

6.297.2.2 `virtual void activemq::commands::ExceptionResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::Response` (p. 2782).

6.297.2.3 `virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p. 2782).

6.297.2.4 `virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Reimplemented from **activemq::commands::Response** (p. 2783).

- 6.297.2.5** `virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ()` [virtual]
- 6.297.2.6** `virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ()` const [virtual]
- 6.297.2.7** `ExceptionResponse& activemq::commands::ExceptionResponse::operator= (const ExceptionResponse &)` [inline, protected]
- 6.297.2.8** `virtual void activemq::commands::ExceptionResponse::setException (const Pointer< BrokerError > & exception)` [virtual]
- 6.297.2.9** `virtual std::string activemq::commands::ExceptionResponse::toString ()` const [virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2783).

6.297.3 Field Documentation

- 6.297.3.1** `Pointer<BrokerError> activemq::commands::ExceptionResponse::exception` [protected]
- 6.297.3.2** `const unsigned char activemq::commands::ExceptionResponse::ID _ - EXCEPTIONRESPONSE = 31` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ExceptionResponse.h`

6.298 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1585).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.298.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1585). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.298.2 Constructor & Destructor Documentation

6.298.2.1 `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionResponseMarshaller()` [inline]

6.298.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~ExceptionResponseMarshaller()` [inline, virtual]

6.298.3 Member Function Documentation

6.298.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.298.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.298.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.298.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.298.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.298.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2794).

6.298.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2795).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h

6.299 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1589).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.299.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1589). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.299.2 Constructor & Destructor Documentation

6.299.2.1 `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionRe
() [inline]`

6.299.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]`

6.299.3 Member Function Documentation

6.299.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.299.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.299.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.299.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2808).

6.299.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2808).

6.299.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2809).

6.299.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2810).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h

6.300 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1593).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.300.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1593). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.300.2 Constructor & Destructor Documentation

6.300.2.1 `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionRe
() [inline]`

6.300.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]`

6.300.3 Member Function Documentation

6.300.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.300.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.300.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.300.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.300.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.300.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2799).

6.300.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2800).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h

6.301 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1597).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.301.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1597). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.301.2 Constructor & Destructor Documentation

6.301.2.1 `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::ExceptionRe
() [inline]`

6.301.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]`

6.301.3 Member Function Documentation

6.301.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.301.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.301.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.301.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.301.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.301.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2814).

6.301.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2815).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h

6.302 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1601).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.302.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1601). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.302.2 Constructor & Destructor Documentation

6.302.2.1 `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::ExceptionRe
() [inline]`

6.302.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]`

6.302.3 Member Function Documentation

6.302.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.302.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.302.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.302.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803).

6.302.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803).

6.302.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2804).

6.302.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h

6.303 decaf::util::concurrent::ExecutionException Class Reference

#include <src/main/decaf/util/concurrent/ExecutionException.h> Inheritance diagram for decaf::util::concurrent::ExecutionException:

Public Member Functions

- **ExecutionException** () throw ()
Default Constructor.
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **ExecutionException** (const **ExecutionException** &ex) throw ()
Copy Constructor.
- **ExecutionException** (const std::exception *cause) throw ()
Constructor.
- **ExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ExecutionException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ExecutionException** * clone () const
Clones this exception.
- virtual ~**ExecutionException** () throw ()

6.303.1 Constructor & Destructor Documentation

6.303.1.1 decaf::util::concurrent::ExecutionException::ExecutionException () throw () [inline]

Default Constructor.

6.303.1.2 decaf::util::concurrent::ExecutionException::ExecutionException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex - An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.303.1.3 `decaf::util::concurrent::ExecutionException::ExecutionException (const ExecutionException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

6.303.1.4 `decaf::util::concurrent::ExecutionException::ExecutionException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.303.1.5 `decaf::util::concurrent::ExecutionException::ExecutionException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - The list of primitives that are formatted into the message

6.303.1.6 `decaf::util::concurrent::ExecutionException::ExecutionException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.303.1.7 virtual decaf::util::concurrent::ExecutionException::~~ExecutionException
() throw () [inline, virtual]

6.303.2 Member Function Documentation

6.303.2.1 virtual ExecutionException* decaf::util::concurrent::ExecutionException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutionException.h**

6.304 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang Runnable** (p. 2816) tasks.

#include <src/main/decaf/util/concurrent/Executor.h> Inheritance diagram for decaf::util::concurrent::Executor:

Public Member Functions

- virtual **~Executor** ()
- virtual void **execute** (Runnable *command)=0 throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException)

Executes the given command at some time in the future.

6.304.1 Detailed Description

An object that executes submitted **decaf.lang Runnable** (p. 2816) tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An **Executor** (p. 1608) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new RunnableTask()).start()` for each of a set of tasks, you might use:

```
Executor (p. 1608) executor = anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the **Executor** (p. 1608) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```
class DirectExecutor : public Executor (p. 1608) {
public:

    void execute( Runnable* r ) (p. 1609) {
        r->run();
    }

}
```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```
class ThreadPerTaskExecutor : public Executor (p. 1608) {
public:
    std::vector<Thread*gt; threads;
```

```
void execute( Runnable* r ) (p.1609) {  
    threads.push_back( new Thread( r ) );  
    threads.rbegin()->start();  
}  
  
}
```

The `Executor` (p.1608) implementations provided in this package implement `decaf.util.concurrent.ExecutorService` (p.1610), which is a more extensive interface. The `decaf.util.concurrent.ThreadPoolExecutor` (p.??) class provides an extensible thread pool implementation. The `decaf.util.concurrent.Executor` (p.??) class provides convenient factory methods for these Executors.

Since:

1.0

6.304.2 Constructor & Destructor Documentation

6.304.2.1 `virtual decaf::util::concurrent::Executor::~~Executor()` [inline, virtual]

6.304.3 Member Function Documentation

6.304.3.1 `virtual void decaf::util::concurrent::Executor::execute(Runnable * command) throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException)` [pure virtual]

Executes the given command at some time in the future. The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the `Executor` (p.1608) implementation.

Parameters:

command the runnable task

Exceptions:

RejectedExecutionException (p. 2699) if this task cannot be accepted for execution.

NullPointerException if command is null

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Executor.h`

6.305 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p.1608) that provides methods to manage termination and methods that can produce a **Future** (p.1701) for tracking progress of one or more asynchronous tasks.

#include <src/main/decaf/util/concurrent/ExecutorService.h> Inheritance diagram for decaf::util::concurrent::ExecutorService:

Public Member Functions

- virtual **~ExecutorService** ()
- bool **awaitTermination** (long long timeout, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException)

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

6.305.1 Detailed Description

An **Executor** (p.1608) that provides methods to manage termination and methods that can produce a **Future** (p.1701) for tracking progress of one or more asynchronous tasks. An **ExecutorService** (p.1610) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p.1610). The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p.1610) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p.1609)(**decaf.lang Runnable** (p.2816)) by creating and returning a **Future** (p.1701) that can be used to cancel execution and/or wait for completion. Methods invokeAny and invokeAll perform the most commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class ExecutorCompletionService can be used to write customized variants of these methods.)

The Executors class provides factory methods for the executor services provided in this package.

Since:

1.0

6.305.2 Constructor & Destructor Documentation

6.305.2.1 `virtual decaf::util::concurrent::ExecutorService::~ExecutorService ()`
[inline, virtual]

6.305.3 Member Function Documentation

6.305.3.1 `bool decaf::util::concurrent::ExecutorService::awaitTermination`
(long long *timeout*, const TimeUnit & *unit*) throw (
decaf::lang::exceptions::InterruptedException) [pure virtual]

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

Parameters:

timeout The amount of time to wait before timing out the Wait operation.

unit The Units that comprise the timeout value.

Returns:

true if the executor terminated before the given timeout value elapsed.

Exceptions:

InterruptedException - if interrupted while waiting.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutorService.h**

6.306 activemq::transport::failover::FailoverTransport Class Reference

#include <src/main/activemq/transport/failover/FailoverTransport.h> Inheritance diagram for activemq::transport::failover::FailoverTransport:

Public Member Functions

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** ()

*Indicates that the **Transport** (p. 3273) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)

Adds a New URI to the List of URIs this transport can Connect to.
- virtual void **addURI** (const **List**< **URI** > &uris)

*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3273) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)

*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3273) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3273) should result in that **Transport** (p. 3273) being disposed of.*
- virtual void **start** () throw (decaf::io::IOException)

Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)

*Stop the **Transport** (p. 3273).*
- virtual void **close** () throw (decaf::io::IOException)

Stops the polling thread and closes the streams.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends the given command to the broker and then waits for the response.

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP_UNUSED)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3273) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3273) been shutdown and no longer usable.*
- bool **isInitialized** () const
*Returns true if the **Transport** (p. 3273) has been initialized by a BrokerInfo command.*
- void **setInitialized** (bool value)
*Sets the initialized state of this **Transport** (p. 3273) to true.*
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()
*Performs the actual Reconnect operation for the **FailoverTransport** (p. 1612), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw (**decaf::io::IOException**)
reconnect to another location
- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const

- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)

Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw (decaf::io::IOException)

*Given a **Transport** (p. 3273) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.*

- void **handleTransportFailure** (const decaf::lang::Exception &error) throw (decaf::lang::Exception)

*Called when this class' **TransportListener** (p. 3289) is notified of a Failure.*

Friends

- class **FailoverTransportListener**

6.306.1 Constructor & Destructor Documentation

6.306.1.1 activemq::transport::failover::FailoverTransport::FailoverTransport ()

6.306.1.2 virtual activemq::transport::failover::FailoverTransport::~~FailoverTransport () [virtual]

6.306.2 Member Function Documentation

6.306.2.1 void activemq::transport::failover::FailoverTransport::add (const std::string & uri)

Adds a New URI to the List of URIs this transport can Connect to.

Parameters:

uri A String version of a URI to add to the URIs to failover to.

6.306.2.2 virtual void activemq::transport::failover::FailoverTransport::addURI (const List< URI > & *uris*) [virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3273) is a composite of.

Parameters:

uris The new URIs to add to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 1095).

6.306.2.3 virtual void activemq::transport::failover::FailoverTransport::close () throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

IOException if errors occur.

Implements **decaf::io::Closeable** (p. 1019).

- 6.306.2.4** `long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier () const [inline]`
- 6.306.2.5** `int activemq::transport::failover::FailoverTransport::getBackupPoolSize () const [inline]`
- 6.306.2.6** `long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay () const [inline]`
- 6.306.2.7** `int activemq::transport::failover::FailoverTransport::getMaxCacheSize () const [inline]`
- 6.306.2.8** `int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts () const [inline]`
- 6.306.2.9** `long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay () const [inline]`
- 6.306.2.10** `long long activemq::transport::failover::FailoverTransport::getReconnectDelay () const [inline]`
- 6.306.2.11** `virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress () const [virtual]`

Returns:

the remote address for this connection

Implements `activemq::transport::Transport` (p. 3274).

- 6.306.2.12** `long long activemq::transport::failover::FailoverTransport::getTimeout () const [inline]`
- 6.306.2.13** `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener () const [virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns:

The listener of transport events.

Implements `activemq::transport::Transport` (p. 3274).

6.306.2.14 void `activemq::transport::failover::FailoverTransport::handleTransportFailure` (const `decaf::lang::Exception & error`) throw (`decaf::lang::Exception`)
[protected]

Called when this class' **TransportListener** (p. 3289) is notified of a Failure.

Parameters:

error - The CMS Exception that was thrown.

Exceptions:

Exception if an error occurs.

6.306.2.15 bool `activemq::transport::failover::FailoverTransport::isBackup` () const
[inline]

6.306.2.16 virtual bool `activemq::transport::failover::FailoverTransport::isClosed` ()
const [inline, virtual]

Has the **Transport** (p. 3273) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 3273)

Implements `activemq::transport::Transport` (p. 3275).

6.306.2.17 virtual bool `activemq::transport::failover::FailoverTransport::isConnected` () const
[inline, virtual]

Is the **Transport** (p. 3273) Connected to its Broker.

Returns:

true if a connection has been made.

Implements `activemq::transport::Transport` (p. 3275).

6.306.2.18 virtual bool `activemq::transport::failover::FailoverTransport::isFaultTolerant` () const
[inline, virtual]

Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 3273) is fault tolerant.

Implements `activemq::transport::Transport` (p. 3275).

6.306.2.19 `bool activemq::transport::failover::FailoverTransport::isInitialized () const [inline]`

Returns true if the **Transport** (p. 3273) has been initialized by a BrokerInfo command.

Returns:

true if the **Transport** (p. 3273) has been initialized by a BrokerInfo command.

6.306.2.20 `virtual bool activemq::transport::failover::FailoverTransport::isPending () const [virtual]`

Returns:

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 1090).

6.306.2.21 `bool activemq::transport::failover::FailoverTransport::isRandomize () const [inline]`

6.306.2.22 `bool activemq::transport::failover::FailoverTransport::isTrackMessages () const [inline]`

6.306.2.23 `bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff () const [inline]`

6.306.2.24 `virtual bool activemq::transport::failover::FailoverTransport::iterate () [virtual]`

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1612), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

Returns:

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 3148).

6.306.2.25 `virtual Transport* activemq::transport::failover::FailoverTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3275).

References **activemq::transport::Transport::narrow()**.

6.306.2.26 **virtual void activemq::transport::failover::FailoverTransport::oneway**
 (const Pointer< Command > & *command*)
 throw (decaf::io::IOException, de-
 caf::lang::exceptions::UnsupportedOperationException)
 [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3276).

6.306.2.27 **virtual void activemq::transport::failover::FailoverTransport::reconnect**
 (const decaf::net::URI & *uri*) throw (decaf::io::IOException)
 [virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

Implements **activemq::transport::Transport** (p. 3276).

6.306.2.28 **void activemq::transport::failover::FailoverTransport::reconnect ()**

Indicates that the **Transport** (p. 3273) needs to reconnect to another URI in its list.

6.306.2.29 **virtual void activemq::transport::failover::FailoverTransport::removeURI**
 (const List< URI > & *uris*) [virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3273) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3273) should result in that **Transport** (p. 3273) being disposed of.

Parameters:

uris The new URIs to remove to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 1096).

6.306.2.30 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters:

command - The command to be sent.
timeout - The time to wait for this response.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.
UnsupportedOperationException if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 3276).

6.306.2.31 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters:

command the command to be sent.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.
UnsupportedOperationException if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 3277).

6.306.2.32 `void activemq::transport::failover::FailoverTransport::restoreTransport (const Pointer< Transport > & transport) throw (decaf::io::IOException) [protected]`

Given a **Transport** (p. 3273) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.

Parameters:

transport The new **Transport** (p. 3273) connected to the Broker.

Exceptions:

IOException if an errors occurs while restoring the old state.

6.306.2.33 void activemq::transport::failover::FailoverTransport::setBackOffMultiplier (long long *value*) [inline]

6.306.2.34 void activemq::transport::failover::FailoverTransport::setBackup (bool *value*) [inline]

6.306.2.35 void activemq::transport::failover::FailoverTransport::setBackupPoolSize (int *value*) [inline]

6.306.2.36 void activemq::transport::failover::FailoverTransport::setInitialized (bool *value*) [inline]

Sets the initialized state of this **Transport** (p. 3273) to true.

Parameters:

value - true if this **Transport** (p. 3273) has been initialized.

- 6.306.2.37 `void activemq::transport::failover::FailoverTransport::setInitialReconnectDelay (long long value) [inline]`
- 6.306.2.38 `void activemq::transport::failover::FailoverTransport::setMaxCacheSize (int value) [inline]`
- 6.306.2.39 `void activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts (int value) [inline]`
- 6.306.2.40 `void activemq::transport::failover::FailoverTransport::setMaxReconnectDelay (long long value) [inline]`
- 6.306.2.41 `void activemq::transport::failover::FailoverTransport::setRandomize (bool value) [inline]`
- 6.306.2.42 `void activemq::transport::failover::FailoverTransport::setReconnectDelay (long long value) [inline]`
- 6.306.2.43 `void activemq::transport::failover::FailoverTransport::setTimeout (long long value) [inline]`
- 6.306.2.44 `void activemq::transport::failover::FailoverTransport::setTrackMessages (bool value) [inline]`
- 6.306.2.45 `virtual void activemq::transport::failover::FailoverTransport::setTransportListener (TransportListener * listener) [virtual]`

Sets the observer of asynchronous events from this transport.

Parameters:

listener the listener of transport events.

Implements `activemq::transport::Transport` (p. 3277).

- 6.306.2.46 `void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff (bool value) [inline]`
- 6.306.2.47 `virtual void activemq::transport::failover::FailoverTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode commands.

6.306.2.48 virtual void activemq::transport::failover::FailoverTransport::start ()
throw (decaf::io::IOException) [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands. If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

IOException if an error occurs or if this transport has already been closed.

Implements **activemq::transport::Transport** (p. 3278).

6.306.2.49 virtual void activemq::transport::failover::FailoverTransport::stop ()
throw (decaf::io::IOException) [virtual]

Stop the **Transport** (p. 3273).

Exceptions:

IOException if an error occurs while stopping the **Transport** (p. 3273).

Implements **activemq::transport::Transport** (p. 3278).

6.306.3 Friends And Related Function Documentation**6.306.3.1** friend class FailoverTransportListener [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransport.h**

6.307 activemq::transport::failover::FailoverTransportFactory Class Reference

Creates an instance of a **FailoverTransport** (p.1612).

#include <src/main/activemq/transport/failover/FailoverTransportFactory.h> Inheritance diagram for activemq::transport::failover::FailoverTransportFactory:

Public Member Functions

- virtual **~FailoverTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a fully configured **Transport** (p.3273) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p.3273) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p.3273) instance which can be used in composite transport instances.*

6.307.1 Detailed Description

Creates an instance of a **FailoverTransport** (p.1612).

Since:

3.0

6.307.2 Constructor & Destructor Documentation

6.307.2.1 virtual
activemq::transport::failover::FailoverTransportFactory::~FailoverTransportFactory
() [inline, virtual]

6.307.3 Member Function Documentation

6.307.3.1 virtual Pointer<Transport> ac-
tivemq::transport::failover::FailoverTransportFactory::create (const
decaf::net::URI & *location*) throw (exceptions::ActiveMQException)
[virtual]

Creates a fully configured **Transport** (p. 3273) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3279).

6.307.3.2 virtual Pointer<Transport> ac-
tivemq::transport::failover::FailoverTransportFactory::createComposite
(const decaf::net::URI & *location*) throw (excep-
tions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3280).

6.307.3.3 virtual Pointer<Transport> ac-
tivemq::transport::failover::FailoverTransportFactory::doCreateComposite
(const decaf::net::URI & *location*, const decaf::util::Properties &
properties) throw (exceptions::ActiveMQException) [protected,
virtual]

Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to.

properties - Properties to apply to the transport.

Returns:

Pointer to a new **FailoverTransport** (p. 1612) instance.

Exceptions:

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

6.308 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 3273) to perform the work of responding to events from the active **Transport** (p. 3273).

#include <src/main/activemq/transport/failover/FailoverTransportListener.h> Inheritance diagram for activemq::transport::failover::FailoverTransportListener:

Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** *parent)
- virtual **~FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

Event handler for the receipt of a command.

- virtual void **onException** (const **decaf::lang::Exception** &ex)

Event handler for an exception from a command transport.

- virtual void **transportInterrupted** ()

The transport has suffered an interruption from which it hopes to recover.

- virtual void **transportResumed** ()

The transport has resumed after an interruption.

6.308.1 Detailed Description

Utility class used by the **Transport** (p. 3273) to perform the work of responding to events from the active **Transport** (p. 3273).

Since:

3.0

6.308.2 Constructor & Destructor Documentation

- 6.308.2.1** `activemq::transport::failover::FailoverTransportListener::FailoverTransportListener(FailoverTransport * parent)`
- 6.308.2.2** `virtual activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener()` [virtual]

6.308.3 Member Function Documentation

- 6.308.3.1** `virtual void activemq::transport::failover::FailoverTransportListener::onCommand(const Pointer< Command > & command)` [virtual]

Event handler for the receipt of a command. The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3273) deletes the command upon receipt.

Parameters:

command the received command object.

Implements `activemq::transport::TransportListener` (p. 3289).

- 6.308.3.2** `virtual void activemq::transport::failover::FailoverTransportListener::onException(const decaf::lang::Exception & ex)` [virtual]

Event handler for an exception from a command transport.

Parameters:

ex The exception.

Implements `activemq::transport::TransportListener` (p. 3290).

- 6.308.3.3** `virtual void activemq::transport::failover::FailoverTransportListener::transportInterrupted()` [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3290).

- 6.308.3.4** `virtual void activemq::transport::failover::FailoverTransportListener::transportResumed()` [virtual]

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3290).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

6.309 decaf::util::logging::Filter Class Reference

A **Filter** (p. 1630) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

Public Member Functions

- virtual **~Filter** ()
- virtual bool **isLoggable** (const **LogRecord** &record) const =0

Check if a given log record should be published.

6.309.1 Detailed Description

A **Filter** (p.1630) can be used to provide fine grain control over what is logged, beyond the control provided by log levels. Each **Logger** (p.2028) and each **Handler** (p.1709) can have a filter associated with it. The **Logger** (p.2028) or **Handler** (p.1709) will call the **isLoggable** method to check if a given **LogRecord** (p.2050) should be published. If **isLoggable** returns false, the **LogRecord** (p.2050) will be discarded.

6.309.2 Constructor & Destructor Documentation

6.309.2.1 virtual decaf::util::logging::Filter::~Filter () [inline, virtual]

6.309.3 Member Function Documentation

6.309.3.1 virtual bool decaf::util::logging::Filter::isLoggable (const **LogRecord** &*record*) const [pure virtual]

Check if a given log record should be published.

Parameters:

record the **LogRecord** (p.2050) to check.

Returns:

true if the record is loggable.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Filter.h**

6.310 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p.1631) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

#include <src/main/decaf/io/FilterInputStream.h> Inheritance diagram for decaf::io::FilterInputStream:

Public Member Functions

- **FilterInputStream** (**InputStream** *inputStream, bool own=false)
*Constructor to create a wrapped **InputStream** (p. 1740).*
- virtual ~**FilterInputStream** ()
- virtual std::size_t **available** () const throw (**IOException**)
Returns the number of bytes that can be read from this input stream without blocking.
- virtual int **read** () throw (**IOException**)
Reads the next byte of data from this input stream.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (**IOException**, lang::exceptions::NullPointerException)
Reads up to len bytes of data from this input stream into an array of bytes.
- virtual void **close** () throw (io::IOException)
*Close the Stream, the **FilterOutputStream** (p. 1640) simply calls the close method of the underlying stream.*
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (**IOException**)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual bool **isClosed** () const

Protected Attributes

- **InputStream * inputStream**
- **util::concurrent::Mutex mutex**
- bool **own**
- volatile bool **closed**

6.310.1 Detailed Description

A **FilterInputStream** (p. 1631) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality. The class **FilterInputStream** (p. 1631) itself simply overrides all methods of **InputStream** (p. 1740) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1631) may further override some of these methods and may also provide additional methods and fields.

6.310.2 Constructor & Destructor Documentation

6.310.2.1 decaf::io::FilterInputStream::FilterInputStream (InputStream * *inputStream*, bool *own* = false) [inline]

Constructor to create a wrapped **InputStream** (p.1740).

Parameters:

inputStream the stream to wrap and filter

own indicates if we own the stream object, defaults to false

6.310.2.2 virtual decaf::io::FilterInputStream::~FilterInputStream () [inline, virtual]

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.310.3 Member Function Documentation

6.310.3.1 virtual std::size_t decaf::io::FilterInputStream::available () const throw (IOException) [inline, virtual]

Returns the number of bytes that can be read from this input stream without blocking. This method simply performs in.available() and returns the result.

Returns:

the number of bytes available without blocking.

Implements **decaf::io::InputStream** (p.1741).

Reimplemented in **decaf::io::BufferedInputStream** (p.810).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.2 virtual void decaf::io::FilterInputStream::close () throw (io::IOException) [inline, virtual]

Close the Stream, the **FilterOutputStream** (p.1640) simply calls the close method of the underlying stream.

Exceptions:

Exception

Implements **decaf::io::Closeable** (p.1019).

Reimplemented in **decaf::io::BufferedInputStream** (p.811).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.3 `virtual bool decaf::io::FilterInputStream::isClosed () const` [inline, protected, virtual]

Returns:

true if this stream has been closed.

6.310.3.4 `virtual void decaf::io::FilterInputStream::lock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.310.3.5 `virtual void decaf::io::FilterInputStream::mark (int readLimit)` [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit The max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1741).

Reimplemented in `decaf::io::BufferedInputStream` (p. 811).

References `DECAF_CATCHALL_NOTHROW`.

6.310.3.6 `virtual bool decaf::io::FilterInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1741).

Reimplemented in `decaf::io::BufferedInputStream` (p. 811).

References `DECAF_CATCHALL_NOTHROW`.

6.310.3.7 virtual void decaf::io::FilterInputStream::notify ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3124).

6.310.3.8 virtual void decaf::io::FilterInputStream::notifyAll ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3125).

6.310.3.9 virtual int decaf::io::FilterInputStream::read (unsigned char * *buffer*,
std::size_t *offset*, std::size_t *bufferSize*) throw (IOException,
lang::exceptions::NullPointerException) [inline, virtual]

Reads up to len bytes of data from this input stream into an array of bytes. This method blocks until some input is available. This method simply performs in.read(b, len) and returns the result.

Parameters:

buffer (out) the target buffer.

offset the position to start reading in the passed buffer.

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF is detected

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException

Implements **decaf::io::InputStream** (p. 1742).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2038), **decaf::io::BufferedInputStream** (p. 811), and **decaf::io::DataInputStream** (p. 1381).

References **DECAF_CATCH_RETHROW**, and **DECAF_CATCHALL_THROW**.

6.310.3.10 virtual int decaf::io::FilterInputStream::read () throw (IOException) [inline, virtual]

Reads the next byte of data from this input stream. The value byte is returned as an unsigned char in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown. This method simply performs `in.read()` and returns the result.

Returns:

The next byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1742).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2039), **decaf::io::BufferedInputStream** (p. 812), and **decaf::io::DataInputStream** (p. 1382).

References **DECAF_CATCH_RETHROW**, and **DECAF_CATCHALL_THROW**.

Referenced by **decaf::io::DataInputStream::read()**.

6.310.3.11 virtual void decaf::io::FilterInputStream::reset () throw (IOException) [inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1820) might be thrown. * If such an **IOException** (p. 1820) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method `markSupported` returns false, then: * The call to reset may throw an **IOException** (p. 1820). * If an **IOException** (p. 1820) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException (p. 1820)

Implements **decaf::io::InputStream** (p. 1742).

Reimplemented in **decaf::io::BufferedInputStream** (p. 812).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.12 `virtual std::size_t decaf::io::FilterInputStream::skip
(std::size_t num) throw (io::IOException,
lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1740) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException (p. 1820) if an error occurs

Implements **decaf::io::InputStream** (p. 1743).

Reimplemented in **decaf::io::BufferedInputStream** (p. 812), and **decaf::io::DataInputStream** (p. 1387).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.13 `virtual bool decaf::io::FilterInputStream::tryLock () throw (
decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.310.3.14 `virtual void decaf::io::FilterInputStream::unlock () throw (
decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.310.3.15 virtual void decaf::io::FilterInputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.310.3.16 virtual void decaf::io::FilterInputStream::wait (long long *millisecs*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.310.3.17 `virtual void decaf::io::FilterInputStream::wait ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

6.310.4 Field Documentation

6.310.4.1 `volatile bool decaf::io::FilterInputStream::closed [protected]`

6.310.4.2 `InputStream* decaf::io::FilterInputStream::inputStream [protected]`

6.310.4.3 `util::concurrent::Mutex decaf::io::FilterInputStream::mutex [protected]`

6.310.4.4 `bool decaf::io::FilterInputStream::own [protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterInputStream.h`

6.311 decaf::io::FilterOutputStream Class Reference

This class is the superclass of all classes that filter output streams.

#include <src/main/decaf/io/FilterOutputStream.h> Inheritance diagram for decaf::io::FilterOutputStream:

Public Member Functions

- **FilterOutputStream** (**OutputStream** *outputStream, bool own=false)
Constructor, creates a wrapped output stream.
- virtual ~**FilterOutputStream** ()
- virtual void **write** (unsigned char c) throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset DECAF_UNUSED, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (IOException)
Flushes any pending writes in this output stream.
- virtual void **close** () throw (io::IOException)
Close the Stream.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual bool **isClosed** () const

Protected Attributes

- **OutputStream * outputStream**
- **util::concurrent::Mutex mutex**
- bool **own**
- volatile bool **closed**

6.311.1 Detailed Description

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1640) itself simply overrides all methods of **OutputStream** (p. 2470) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1640) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1740) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataOutputStream (p. 1388) os = new DataOutputStream (p. 1388)( new OutputStream(), true )
```

6.311.2 Constructor & Destructor Documentation

6.311.2.1 decaf::io::FilterOutputStream::FilterOutputStream (OutputStream * outputStream, bool own = false) [inline]

Constructor, creates a wrapped output stream.

Parameters:

outputStream the **OutputStream** (p. 2470) to wrap

own If true, this object will control the lifetime of the output stream that it encapsulates.

6.311.2.2 `virtual decaf::io::FilterOutputStream::~~FilterOutputStream () [inline, virtual]`

References `DECAF_CATCH_NOTHROW`, and `DECAF_CATCHALL_NOTHROW`.

6.311.3 Member Function Documentation

6.311.3.1 `virtual void decaf::io::FilterOutputStream::close () throw (io::IOException) [inline, virtual]`

Close the Stream. The close method of **FilterOutputStream** (p. 1640) calls its flush method, and then calls the close method of its underlying output stream, it then destroys the output stream if it is the owner.

Exceptions:

Exception

Implements **decaf::io::Closeable** (p. 1019).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 815).

References `DECAF_CATCH_RETHROW`, and `DECAF_CATCHALL_THROW`.

6.311.3.2 `virtual void decaf::io::FilterOutputStream::flush () throw (IOException) [inline, virtual]`

Flushes any pending writes in this output stream. The flush method of **FilterOutputStream** (p. 1640) calls the flush method of its underlying output stream

Exceptions:

IOException (p. 1820)

Implements **decaf::io::OutputStream** (p. 2470).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 815).

References `DECAF_CATCH_RETHROW`, and `DECAF_CATCHALL_THROW`.

6.311.3.3 `virtual bool decaf::io::FilterOutputStream::isClosed () const [inline, protected, virtual]`

Returns:

true if this stream has been closed.

6.311.3.4 virtual void decaf::io::FilterOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 3123).

6.311.3.5 virtual void decaf::io::FilterOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3124).

6.311.3.6 virtual void decaf::io::FilterOutputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3125).

6.311.3.7 virtual bool decaf::io::FilterOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3126).

6.311.3.8 `virtual void decaf::io::FilterOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3127).

6.311.3.9 `virtual void decaf::io::FilterOutputStream::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3128).

6.311.3.10 `virtual void decaf::io::FilterOutputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.311.3.11 virtual void decaf::io::FilterOutputStream::wait ()
 throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

6.311.3.12 virtual void decaf::io::FilterOutputStream::write (const unsigned char *
buffer, std::size_t offset *DECAF_UNUSED*, std::size_t *len*) throw
 (IOException, lang::exceptions::NullPointerException) [inline,
 virtual]

Writes an array of bytes to the output stream. The write method of **FilterOutputStream** (p. 1640) calls the write method of one argument on each byte to output.

Parameters:

buffer The array of bytes to write.

offset, the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 2471).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 2040), **decaf::io::BufferedOutputStream** (p. 815), and **decaf::io::DataOutputStream** (p. 1390).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.311.3.13 `virtual void decaf::io::FilterOutputStream::write (const std::vector< unsigned char > & buffer) throw (IOException)` [inline, virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements `decaf::io::OutputStream` (p. 2471).

Reimplemented in `decaf::io::BufferedOutputStream` (p. 816), and `decaf::io::DataOutputStream` (p. 1390).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.311.3.14 `virtual void decaf::io::FilterOutputStream::write (unsigned char c) throw (IOException)` [inline, virtual]

Writes a single byte to the output stream. The write method of `FilterOutputStream` (p. 1640) calls the write method of its underlying output stream, that is, it performs `out.write(b)`.

Parameters:

c the byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implements `decaf::io::OutputStream` (p. 2471).

Reimplemented in `activemq::io::LoggingOutputStream` (p. 2041), `decaf::io::BufferedOutputStream` (p. 816), and `decaf::io::DataOutputStream` (p. 1390).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.311.4 Field Documentation

6.311.4.1 `volatile bool decaf::io::FilterOutputStream::closed` [protected]

6.311.4.2 `util::concurrent::Mutex decaf::io::FilterOutputStream::mutex` [protected]

6.311.4.3 `OutputStream* decaf::io::FilterOutputStream::outputStream` [protected]

6.311.4.4 `bool decaf::io::FilterOutputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterOutputStream.h`

6.312 decaf::lang::Float Class Reference

#include <src/main/decaf/lang/Float.h> Inheritance diagram for decaf::lang::Float:

Public Member Functions

- **Float** (float value)
- **Float** (double value)
- **Float** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const
*Compares this **Float** (p. 1647) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Float** &f) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const float &f) const
*Compares this **Float** (p. 1647) instance with another.*
- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const float &f) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.

- virtual long long **longValue** () const
Answers the long value which the receiver represents.
- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (float f1, float f2)
Compares the two specified double values.
- static int **floatToIntBits** (float value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.
- static int **floatToRawIntBits** (float value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.
- static float **intBitsToFloat** (int bits)
Returns the float value corresponding to a given bit representation.
- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1647).*
- static std::string **toHexString** (float value)
Returns a hexadecimal string representation of the float argument.
- static std::string **toString** (float value)
Returns a string representation of the float argument.
- static **Float** **valueOf** (float value)
*Returns a **Float** (p. 1647) instance representing the specified float value.*
- static **Float** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Float** (p. 1647) instance that wraps a primitive float which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const float **MAX_VALUE**

The maximum value that the primitive type can hold.

- static const float **MIN_VALUE**

The minimum value that the primitive type can hold.

- static const float **NaN**

*Constant for the Not a **Number** (p. 2432) Value.*

- static const float **POSITIVE_INFINITY**

Constant for Positive Infinity.

- static const float **NEGATIVE_INFINITY**

Constant for Negative Infinity.

6.312.1 Constructor & Destructor Documentation

6.312.1.1 decaf::lang::Float::Float (float *value*)

Parameters:

value - the primitive type to wrap

6.312.1.2 decaf::lang::Float::Float (double *value*)

Parameters:

value - the primitive type to wrap

6.312.1.3 decaf::lang::Float::Float (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - the string to convert to a primitive type to wrap

6.312.1.4 virtual decaf::lang::Float::~~Float () [inline, virtual]

6.312.2 Member Function Documentation

6.312.2.1 virtual unsigned char decaf::lang::Float::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2432).

6.312.2.2 static int decaf::lang::Float::compare (float *f1*, float *f2*) [static]

Compares the two specified double values. The sign of the integer value returned is the same as that of the integer that would be returned by the call: `Float(f1).compareTo(Float(f2))`

Parameters:

f1 - the first double to compare

f2 - the second double to compare

Returns:

the value 0 if *d1* is numerically equal to *f2*; a value less than 0 if *f1* is numerically less than *f2*; and a value greater than 0 if *f1* is numerically greater than *f2*.

6.312.2.3 virtual int decaf::lang::Float::compareTo (const float & *f*) const [virtual]

Compares this **Float** (p. 1647) instance with another.

Parameters:

f - the **Float** (p. 1647) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< float >** (p. 1083).

6.312.2.4 virtual int decaf::lang::Float::compareTo (const Float & *f*) const [virtual]

Compares this **Float** (p. 1647) instance with another.

Parameters:

f - the **Float** (p. 1647) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< Float >** (p. 1083).

6.312.2.5 virtual double decaf::lang::Float::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.312.2.6 `bool decaf::lang::Float::equals (const float & f) const` [inline, virtual]**Parameters:**

f - the **Float** (p. 1647) object to compare against.

Returns:

true if the two **Float** (p. 1647) Objects have the same value.

Implements **decaf::lang::Comparable**< float > (p. 1084).

6.312.2.7 `bool decaf::lang::Float::equals (const Float & f) const` [inline, virtual]**Parameters:**

f - the **Float** (p. 1647) object to compare against.

Returns:

true if the two **Float** (p. 1647) Objects have the same value.

Implements **decaf::lang::Comparable**< Float > (p. 1084).

6.312.2.8 `static int decaf::lang::Float::floatToIntBits (float value)` [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout. Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7fc00000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1652) method, will produce a floating-point value the same as the argument to floatToIntBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters:

value - the float to convert to int bits

Returns:

the int that holds the float's value

6.312.2.9 static int decaf::lang::Float::floatToRawIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values. Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the floatToIntBits method, intToRawIntBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1652) method, will produce a floating-point value the same as the argument to floatToRawIntBits.

Parameters:

value The float to convert to a raw int.

Returns:

the raw int value of the float

6.312.2.10 virtual float decaf::lang::Float::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.312.2.11 static float decaf::lang::Float::intBitsToFloat (int *bits*) [static]

Returns the float value corresponding to a given bit representation. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1652) method.

Parameters:

bits - the bits of the float encoded as a float

Returns:

a new float created from the int bits.

6.312.2.12 `virtual int decaf::lang::Float::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.312.2.13 `static bool decaf::lang::Float::isInfinite (float value) [static]`**Parameters:**

value - The float to check.

Returns:

true if the float is equal to infinity.

6.312.2.14 `bool decaf::lang::Float::isInfinite () const`**Returns:**

true if the float is equal to positive infinity.

6.312.2.15 `static bool decaf::lang::Float::isNaN (float value) [static]`**Parameters:**

value - The float to check.

Returns:

true if the float is equal to NaN.

6.312.2.16 `bool decaf::lang::Float::isNaN () const`**Returns:**

true if the float is equal to NaN.

6.312.2.17 `virtual long long decaf::lang::Float::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.312.2.18 `virtual bool decaf::lang::Float::operator< (const float & f) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< float >` (p.1084).

6.312.2.19 `virtual bool decaf::lang::Float::operator< (const Float & f) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< Float >` (p.1084).

6.312.2.20 `virtual bool decaf::lang::Float::operator== (const float & f) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< float >` (p.1085).

6.312.2.21 `virtual bool decaf::lang::Float::operator== (const Float & f) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Float** > (p.1085).

6.312.2.22 static float decaf::lang::Float::parseFloat (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p.1647).

Parameters:

value - the string to parse

Returns:

a float parsed from the string

Exceptions:

NumberFormatException

6.312.2.23 virtual short decaf::lang::Float::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p.2434).

6.312.2.24 static std::string decaf::lang::Float::toHexString (float *value*) [static]

Returns a hexadecimal string representation of the float argument. All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p.1774) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the

hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters:

value - The float to convert to a string

Returns:

the Hex formatted float string.

6.312.2.25 static std::string decaf::lang::Float::toString (float *value*) [static]

Returns a string representation of the float argument. All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude *m*:
 o If *m* is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".
 o If *m* is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".
 o If *m* is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of *m*, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of *m*.
 o If *m* is less than 10⁻³ or greater than or equal to 10⁷, then it is represented in so-called "computerized scientific notation." Let *n* be the unique integer such that 10^{*n*} ≤ *m* < 10^{*n*+1}; then let *a* be the mathematically exact quotient of *m* and 10^{*n*} so that 1 ≤ *a* < 10. The magnitude is then represented as the integer part of *a*, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of *a*, followed by the letter 'E', followed by a representation of *n* as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1774).

Parameters:

value - The float to convert to a string

Returns:

the formatted float string.

6.312.2.26 std::string decaf::lang::Float::toString () const

Returns:

this **Float** (p. 1647) Object as a String Representation

6.312.2.27 static Float decaf::lang::Float::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Float** (p. 1647) instance that wraps a primitive float which is parsed from the string value passed.

Parameters:

value - the string to parse

Returns:

a new **Float** (p. 1647) instance wrapping the float parsed from value

Exceptions:

NumberFormatException on error.

6.312.2.28 static Float decaf::lang::Float::valueOf (float value) [static]

Returns a **Float** (p. 1647) instance representing the specified float value.

Parameters:

value - float to wrap

Returns:

new **Float** (p. 1647) instance wrapping the primitive value

6.312.3 Field Documentation**6.312.3.1 const float decaf::lang::Float::MAX_VALUE [static]**

The maximum value that the primitive type can hold.

6.312.3.2 const float decaf::lang::Float::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.312.3.3 const float decaf::lang::Float::NaN [static]

Constant for the Not a **Number** (p. 2432) Value.

6.312.3.4 const float decaf::lang::Float::NEGATIVE_INFINITY [static]

Constant for Negative Infinity.

6.312.3.5 const float decaf::lang::Float::POSITIVE_INFINITY [static]

Constant for Positive Infinity.

6.312.3.6 const int decaf::lang::Float::SIZE = 32 [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Float.h**

6.313 decaf::internal::nio::FloatArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/FloatArrayBuffer.h> Inheritance diagram for decaf::internal::nio::FloatArrayBuffer:

Public Member Functions

- **FloatArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **FloatArrayBuffer** (p. 1658) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **FloatArrayBuffer** (float *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **FloatArrayBuffer** (p. 1658) object that wraps the given array.*
- **FloatArrayBuffer** (ByteBufferPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteBufferPerspective** (p. 908) and start at the given offset.*
- **FloatArrayBuffer** (const **FloatArrayBuffer** &other)
*Create a **FloatArrayBuffer** (p. 1658) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~**FloatArrayBuffer** ()
- virtual float * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the float array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual FloatBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only float buffer that shares this buffer's content.
- virtual FloatBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual FloatBuffer * **duplicate** ()
Creates a new float buffer that shares this buffer's content.
- virtual float **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual float **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible float array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual FloatBuffer & **put** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual FloatBuffer & **put** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual FloatBuffer * **slice** () const
Creates a new FloatBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteBuffer** (p. 870) as Read-Only.*

6.313.1 Constructor & Destructor Documentation

6.313.1.1 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **FloatArrayBuffer** (p. 1658) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.
readOnly - should this buffer be read-only, default as false

6.313.1.2 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **FloatArrayBuffer** (p. 1658) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.313.1.3 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer` (`ByteArrayPerspective & array`, `std::size_t offset`, `std::size_t capacity`, `bool readOnly = false`) `throw (` `decaf::lang::exceptions::IndexOutOfBoundsException` `)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p.908) and start at the given offset. The capacity and limit of the new **FloatArrayBuffer** (p.1658) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p.908) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.313.1.4 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer` (`const` `FloatArrayBuffer & other`)

Create a **FloatArrayBuffer** (p.1658) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p.908) and when changes are made to that data it is reflected in both.

Parameters:

other - the **FloatArrayBuffer** (p.1658) this one is to mirror.

6.313.1.5 `virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer ()` `[virtual]`

6.313.2 Member Function Documentation

6.313.2.1 `virtual float* decaf::internal::nio::FloatArrayBuffer::array ()` `throw (decaf::lang::exceptions::UnsupportedOperationException,` `decaf::nio::ReadOnlyBufferException)` `[virtual]`

Returns the float array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p.1667).

6.313.2.2 `virtual std::size_t decaf::internal::nio::FloatArrayBuffer::arrayOffset
() throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p.1668).

6.313.2.3 `virtual FloatBuffer* de-
caf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer ()
const [virtual]`

Creates a new, read-only float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p.1668).

6.313.2.4 `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p.807) is copied

to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index **limit()** (p. 807) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this FloatBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1668).

6.313.2.5 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::duplicate ()
[virtual]

Creates a new float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new float Buffer which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1669).

6.313.2.6 virtual float decaf::internal::nio::FloatArrayBuffer::get (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the float is to be read

Returns:

the float that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::FloatBuffer** (p. 1670).

6.313.2.7 virtual float decaf::internal::nio::FloatArrayBuffer::get () throw (
decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the float at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::FloatBuffer** (p.1671).

6.313.2.8 virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible float array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::FloatBuffer** (p.1671).

6.313.2.9 virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 806).

6.313.2.10 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p.1672).

6.313.2.11 `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::FloatBuffer` (p.1672).

6.313.2.12 `virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this `ByteBuffer` (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.313.2.13 `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice () const` [virtual]

Creates a new `FloatBuffer` whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create `FloatBuffer` which the caller owns.

Implements `decaf::nio::FloatBuffer` (p.1674).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/FloatArrayBuffer.h`

6.314 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:.

#include <src/main/decaf/nio/FloatBuffer.h> Inheritance diagram for decaf::nio::FloatBuffer:

Public Member Functions

- virtual **~FloatBuffer** ()
- virtual std::string **toString** () const
- virtual float * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the float array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **FloatBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only float buffer that shares this buffer's content.
- virtual **FloatBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **FloatBuffer** * **duplicate** ()=0
Creates a new float buffer that shares this buffer's content.
- virtual float **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual float **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **FloatBuffer** & **get** (std::vector< float > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **FloatBuffer** & **get** (float *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible float array.
- **FloatBuffer** & **put** (**FloatBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the floats remaining in the given source buffer into this buffer.

- **FloatBuffer** & **put** (const float *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers floats into this buffer from the given source array.
- **FloatBuffer** & **put** (std::vector< float > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source floats array into this buffer.
- virtual **FloatBuffer** & **put** (float value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given floats into this buffer at the current position, and then increments the position.
- virtual **FloatBuffer** & **put** (std::size_t index, float value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given floats into this buffer at the given index.
- virtual **FloatBuffer** * **slice** () const =0
*Creates a new **FloatBuffer** (p. 1665) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **FloatBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **FloatBuffer** &value) const
- virtual bool **operator==** (const **FloatBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **FloatBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **FloatBuffer** * **allocate** (std::size_t capacity)
Allocates a new Double buffer.
- static **FloatBuffer** * **wrap** (float *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **FloatBuffer** (p. 1665).*
- static **FloatBuffer** * **wrap** (std::vector< float > &buffer)
*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1665).*

Protected Member Functions

- **FloatBuffer** (std::size_t capacity)
*Creates a **FloatBuffer** (p. 1665) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.314.1 Detailed Description

This class defines four categories of operations upon float buffers:.

- o Absolute and relative get and put methods that read and write single floats;
- o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and
- o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer
- o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.314.2 Constructor & Destructor Documentation

6.314.2.1 decaf::nio::FloatBuffer::FloatBuffer (std::size_t *capacity*) [protected]

Creates a **FloatBuffer** (p. 1665) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 803) in doubles

6.314.2.2 virtual decaf::nio::FloatBuffer::~~FloatBuffer () [inline, virtual]

6.314.3 Member Function Documentation

6.314.3.1 static FloatBuffer* decaf::nio::FloatBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in floats

Returns:

the **FloatBuffer** (p. 1665) that was allocated, caller owns.

6.314.3.2 virtual float* decaf::nio::FloatBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the float array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 803)

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1660).

6.314.3.3 `virtual std::size_t decaf::nio::FloatBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1661).

6.314.3.4 `virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1661).

6.314.3.5 `virtual FloatBuffer& decaf::nio::FloatBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]`

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 807) is copied

to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index `limit()` (p. 807) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **FloatBuffer** (p. 1665)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1661).

6.314.3.6 virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.314.3.7 virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate () [pure virtual]

Creates a new float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new float **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1662).

6.314.3.8 virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.314.3.9 FloatBuffer& decaf::nio::FloatBuffer::get (float * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies `length` floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters:

- buffer* - pointer to an allocated buffer to fill
- offset* - position in the buffer to start filling
- length* - amount of data to put in the passed buffer

Returns:

- a reference to this **Buffer** (p. 803)

Exceptions:

- BufferUnderflowException** (p. 837) - If there are fewer than `length` floats remaining in this buffer
- NullPointerException** if the passed buffer is null.

6.314.3.10 FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns:

- a reference to this **Buffer** (p. 803)

Exceptions:

- BufferUnderflowException** (p. 837) - If there are fewer than `length` floats remaining in this buffer

6.314.3.11 virtual float decaf::nio::FloatBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the value at the given index.

Parameters:

- index* - the index in the **Buffer** (p. 803) where the float is to be read

Returns:

the float that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1662).

6.314.3.12 virtual float decaf::nio::FloatBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the float at the current position

Exceptions:

BufferUnderflowException (p. 837) if there no more data to return

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1662).

6.314.3.13 virtual bool decaf::nio::FloatBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible float array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1663).

6.314.3.14 virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer &value) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.314.3.15 `virtual bool decaf::nio::FloatBuffer::operator==(const FloatBuffer & value) const` [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.314.3.16 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given floats into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the floats to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1663).

6.314.3.17 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (float value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters:

value - the floats value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1664).

6.314.3.18 FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source floats array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size())

Parameters:

buffer - The buffer whose contents are copied to this **FloatBuffer** (p. 1665)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.314.3.19 FloatBuffer& decaf::nio::FloatBuffer::put (const float * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers floats into this buffer from the given source array. If there are more floats to be copied from the array than remain in this buffer, that is, if *length* > **remaining()** (p. 808), then no floats are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies *length* bytes from the given array into this buffer, starting at the given *offset* in the array and at the current position of this buffer. The position of this buffer is then incremented by *length*.

Parameters:

buffer- The array from which floats are to be read

offset- The offset within the array of the first float to be read;

length - The number of floats to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.314.3.20 FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the floats remaining in the given source buffer into this buffer. If there are more floats remaining in the source buffer than in this buffer, that is, if *src.remaining()* >

remaining() (p. 808), then no floats are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies $n = \text{src.remaining}()$ floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n .

Parameters:

src - the buffer to take floats from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining floats in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.314.3.21 `virtual FloatBuffer* decaf::nio::FloatBuffer::slice () const` [pure virtual]

Creates a new **FloatBuffer** (p. 1665) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **FloatBuffer** (p. 1665) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1664).

6.314.3.22 `virtual std::string decaf::nio::FloatBuffer::toString () const` [virtual]

Returns:

a `std::string` describing this object

6.314.3.23 `static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & buffer)` [static]

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1665). The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new **FloatBuffer** (p. 1665) that is backed by *buffer*, caller owns.

6.314.3.24 `static FloatBuffer* decaf::nio::FloatBuffer::wrap (float *
array, std::size_t offset, std::size_t length) throw (
lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **FloatBuffer** (p. 1665). The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **FloatBuffer** (p. 1665) that is backed by *buffer*, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/FloatBuffer.h`

6.315 activemq::commands::FlushCommand Class Reference

#include <src/main/activemq/commands/FlushCommand.h> Inheritance diagram for activemq::commands::FlushCommand:

Public Member Functions

- **FlushCommand** ()
- virtual **~FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **FlushCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _FLUSHCOMMAND** = 15

Protected Member Functions

- **FlushCommand** (const **FlushCommand** &)
- **FlushCommand & operator=** (const **FlushCommand** &)

6.315.1 Constructor & Destructor Documentation

6.315.1.1 `activemq::commands::FlushCommand::FlushCommand (const FlushCommand &) [inline, protected]`

6.315.1.2 `activemq::commands::FlushCommand::FlushCommand ()`

6.315.1.3 `virtual activemq::commands::FlushCommand::~~FlushCommand () [virtual]`

6.315.2 Member Function Documentation

6.315.2.1 `virtual FlushCommand* activemq::commands::FlushCommand::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.315.2.2 `virtual void activemq::commands::FlushCommand::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.315.2.3 `virtual bool activemq::commands::FlushCommand::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.315.2.4 `virtual unsigned char activemq::commands::FlushCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

6.315.2.5 **FlushCommand& activemq::commands::FlushCommand::operator=**
(const **FlushCommand** &) [inline, protected]

6.315.2.6 **virtual std::string activemq::commands::FlushCommand::toString ()**
const [virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.315.2.7 **virtual Pointer<Command> activemq::commands::FlushCommand::visit**
(activemq::state::CommandVisitor * *visitor*) throw (
exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.315.3 Field Documentation

6.315.3.1 **const unsigned char activemq::commands::FlushCommand::ID_ -**
FLUSHCOMMAND = 15 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**FlushCommand.h**

6.316 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1679).

#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.316.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1679). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.316.2 Constructor & Destructor Documentation

6.316.2.1 `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.316.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.316.3 Member Function Documentation

6.316.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.316.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.316.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 686).

6.316.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 687).

6.316.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 688).

6.316.3.6 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.316.3.7 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h

6.317 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1683).

#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.317.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1683). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.317.2 Constructor & Destructor Documentation

6.317.2.1 `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

6.317.2.2 `virtual activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

6.317.3 Member Function Documentation

6.317.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.317.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.317.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 665).

6.317.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 666).

6.317.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 667).

6.317.3.6 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.317.3.7 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**

6.318 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1687).

#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.318.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1687). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.318.2 Constructor & Destructor Documentation

6.318.2.1 `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

6.318.2.2 `virtual activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

6.318.3 Member Function Documentation

6.318.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.318.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.318.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.318.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.318.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.318.3.6 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.318.3.7 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h

6.319 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1691).

#include <src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.319.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1691). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.319.2 Constructor & Destructor Documentation

6.319.2.1 `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::FlushCommandMarshaller()` [inline]

6.319.2.2 `virtual activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::~~FlushCommandMarshaller()` [inline, virtual]

6.319.3 Member Function Documentation

6.319.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.319.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.319.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 672).

6.319.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 673).

6.319.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 674).

6.319.3.6 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.319.3.7 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h

6.320 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1695).

#include <src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.320.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1695). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.320.2 Constructor & Destructor Documentation

6.320.2.1 `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::FlushCommandM`
`() [inline]`

6.320.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::~~FlushCommand`
`() [inline, virtual]`

6.320.3 Member Function Documentation

6.320.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.320.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.320.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 679).

6.320.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 680).

6.320.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 681).

6.320.3.6 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.320.3.7 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h

6.321 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1699) provides support for formatting LogRecords.

#include <src/main/decaf/util/logging/Formatter.h> Inheritance diagram for decaf::util::logging::Formatter:

Public Member Functions

- virtual **~Formatter** ()
- virtual std::string **format** (const **LogRecord** &record) const =0
Format the given log record and return the formatted string.
- virtual std::string **formatMessage** (const **LogRecord** &record) const =0
Format the message string from a log record.
- virtual std::string **getHead** (const **Handler** *handler)=0
Return the header string for a set of formatted records.
- virtual std::string **getTail** (const **Handler** *handler)=0
Return the tail string for a set of formatted records.

6.321.1 Detailed Description

A **Formatter** (p. 1699) provides support for formatting LogRecords. Typically each logging **Handler** (p. 1709) will have a **Formatter** (p. 1699) associated with it. The **Formatter** (p. 1699) takes a **LogRecord** (p. 2050) and converts it to a string.

Some formatters (such as the XMLFormatter) need to wrap head and tail strings around a set of formatted records. The getHeader and getTail methods can be used to obtain these strings.

6.321.2 Constructor & Destructor Documentation

6.321.2.1 virtual decaf::util::logging::Formatter::~~Formatter () [inline, virtual]

6.321.3 Member Function Documentation

6.321.3.1 virtual std::string decaf::util::logging::Formatter::format (const **LogRecord** & *record*) const [pure virtual]

Format the given log record and return the formatted string.

Parameters:

record The Log Record to Format

Returns:

the formatted record.

Implemented in **decaf::util::logging::SimpleFormatter** (p. 2960).

6.321.3.2 `virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & record) const` [pure virtual]

Format the message string from a log record.

Parameters:

record The Log Record to Format

Returns:

the formatted message

Implemented in `decaf::util::logging::SimpleFormatter` (p. 2961).

6.321.3.3 `virtual std::string decaf::util::logging::Formatter::getHead (const Handler * handler)` [pure virtual]

Return the header string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

the head string

Implemented in `decaf::util::logging::SimpleFormatter` (p. 2961).

6.321.3.4 `virtual std::string decaf::util::logging::Formatter::getTail (const Handler * handler)` [pure virtual]

Return the tail string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

the tail string

Implemented in `decaf::util::logging::SimpleFormatter` (p. 2961).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Formatter.h`

6.322 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p. 1701) represents the result of an asynchronous computation.

```
#include <src/main/decaf/util/concurrent/Future.h>
```

Public Member Functions

- virtual **~Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0
Attempts to cancel execution of this task.
- bool **isCancelled** () const =0
Returns true if this task was canceled before it completed normally.
- bool **isDone** () const =0
Returns true if this task completed.
- V **get** ()=0 throw (CancellationException, InterruptedException, ExecutionException)
Waits if necessary for the computation to complete, and then retrieves its result.
- V **get** (long long timeout, **TimeUnit** unit)=0 throw (InterruptedException, ExecutionException, TimeoutException)
Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

6.322.1 Detailed Description

```
template<typename V> class decaf::util::concurrent::Future< V >
```

A **Future** (p. 1701) represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method **get** when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the **cancel** method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p. 1701) for the sake of cancellability but not provide a usable result, you can declare types of the form `Future<void*>` and return null as a result of the underlying task.

6.322.2 Constructor & Destructor Documentation

6.322.2.1 `template<typename V > virtual decaf::util::concurrent::Future< V >::~~Future () [inline, virtual]`

6.322.3 Member Function Documentation

6.322.3.1 `template<typename V > bool decaf::util::concurrent::Future< V >::cancel (bool mayInterruptIfRunning) [pure virtual]`

Attempts to cancel execution of this task. This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when cancel is called, this task should never run. If the task has already started, then the *mayInterruptIfRunning* parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to **isDone()** (p. 1703) will always return true. Subsequent calls to **isCancelled()** (p. 1703) will always return true if this method returned true.

Parameters:

mayInterruptIfRunning - true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.

Returns:

false if the task could not be canceled, typically because it has already completed normally;
true otherwise

6.322.3.2 `template<typename V > V decaf::util::concurrent::Future< V >::get (long long timeout, TimeUnit unit) throw (InterruptedException, ExecutionException, TimeoutException) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

Parameters:

timeout - the maximum time to wait

unit - the time unit of the timeout argument

Returns:

the computed result

Exceptions:

CancellationException (p. 965) - if the computation was canceled

ExecutionException (p. 1605) - if the computation threw an exception

InterruptedException - if the current thread was interrupted while waiting

TimeoutException (p. 3185) - if the wait timed out

6.322.3.3 `template<typename V > V decaf::util::concurrent::Future< V >::get () throw (CancellationException, InterruptedException, ExecutionException) [pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

Returns:

the computed result.

Exceptions:

CancellationException (p. 965) - if the computation was canceled
ExecutionException (p. 1605) - if the computation threw an exception
InterruptedException - if the current thread was interrupted while waiting

6.322.3.4 `template<typename V > bool decaf::util::concurrent::Future< V >::isCancelled () const [pure virtual]`

Returns true if this task was canceled before it completed normally.

Returns:

true if this task was canceled before it completed

6.322.3.5 `template<typename V > bool decaf::util::concurrent::Future< V >::isDone () const [pure virtual]`

Returns true if this task completed. Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

Returns:

true if this task completed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/`**Future.h**

6.323 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/FutureResponse.h>
```

Public Member Functions

- **FutureResponse** ()
- virtual **~FutureResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** () const
Getters for the response property.
- virtual **Pointer**< **Response** > & **getResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** (unsigned int timeout) const
Getters for the response property.
- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout)
- virtual void **setResponse** (const **Pointer**< **Response** > &response)
Setter for the response property.

6.323.1 Detailed Description

A container that holds a response object. Callers of the `getResponse` method will block until a response has been receive unless they call the `getRepsonse` that takes a timeout.

6.323.2 Constructor & Destructor Documentation

6.323.2.1 `activemq::transport::correlator::FutureResponse::FutureResponse ()`
[inline]

6.323.2.2 `virtual`
`activemq::transport::correlator::FutureResponse::~~FutureResponse ()`
[inline, virtual]

6.323.3 Member Function Documentation

6.323.3.1 `virtual` **Pointer**<**Response**>& `activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout)` [inline, virtual]

6.323.3.2 `virtual` const **Pointer**<**Response**>& `activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout)` const [inline, virtual]

Getters for the response property. Timed Wait.

Parameters:

timeout - time to wait in milliseconds

Returns:

the response object for the request

6.323.3.3 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () [inline, virtual]`

6.323.3.4 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () const [inline, virtual]`

Getters for the response property. Infinite Wait.

Returns:

the response object for the request

6.323.3.5 `virtual void activemq::transport::correlator::FutureResponse::setResponse (const Pointer< Response > & response) [inline, virtual]`

Setter for the response property.

Parameters:

response the response object for the request.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/correlator/FutureResponse.h`

6.324 decaf::security::GeneralSecurityException Class Reference

#include <src/main/decaf/security/GeneralSecurityException.h> Inheritance diagram for decaf::security::GeneralSecurityException:

Public Member Functions

- **GeneralSecurityException** () throw ()
Default Constructor.
- **GeneralSecurityException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **GeneralSecurityException** (const **GeneralSecurityException** &ex) throw ()
Copy Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **GeneralSecurityException** (const std::exception *cause) throw ()
Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **GeneralSecurityException** * clone () const
Clones this exception.
- virtual ~**GeneralSecurityException** () throw ()

6.324.1 Constructor & Destructor Documentation

6.324.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException () throw () [inline]

Default Constructor.

6.324.1.2 decaf::security::GeneralSecurityException::GeneralSecurityException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.324.1.3 decaf::security::GeneralSecurityException::GeneralSecurityException (const GeneralSecurityException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.324.1.4 decaf::security::GeneralSecurityException::GeneralSecurityException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.324.1.5 decaf::security::GeneralSecurityException::GeneralSecurityException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.324.1.6 decaf::security::GeneralSecurityException::GeneralSecurityException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.324.1.7 `virtual`
`decaf::security::GeneralSecurityException::~~GeneralSecurityException ()`
`throw ()` [`inline`, `virtual`]

6.324.2 Member Function Documentation

6.324.2.1 `virtual GeneralSecurityException* decaf::security::GeneralSecurityException::clone () const`
`[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 973), `decaf::security::cert::CertificateException` (p. 975), `decaf::security::cert::CertificateExpiredException` (p. 977), `decaf::security::cert::CertificateNotYetValidException` (p. 979), `decaf::security::cert::CertificateParsingException` (p. 981), `decaf::security::InvalidKeyException` (p. 1812), `decaf::security::KeyException` (p. 1957), `decaf::security::NoSuchAlgorithmException` (p. 2422), `decaf::security::NoSuchProviderException` (p. 2428), and `decaf::security::SignatureException` (p. 2959).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/GeneralSecurityException.h`

6.325 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1709) object takes log messages from a **Logger** (p. 2028) and exports them.

#include <src/main/decaf/util/logging/Handler.h> Inheritance diagram for decaf::util::logging::Handler:

Public Member Functions

- virtual **~Handler** ()
- virtual void **flush** ()=0
Flush the Handler's output, clears any buffers.
- virtual void **publish** (const **LogRecord** &record)=0
*Publish the Log Record to this **Handler** (p. 1709).*
- virtual void **isLoggable** (const **LogRecord** &record)=0
*Check if this **Handler** (p. 1709) would actually log a given **LogRecord** (p. 2050).*
- virtual void **setFilter** (const **Filter** *filter)=0
*Sets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.*
- virtual const **Filter** * **getFilter** ()=0
*Gets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.*
- virtual void **setLevel** (**Level** value)=0
***Set** (p. 2905) the log level specifying which message levels will be logged by this **Handler** (p. 1709).*
- virtual **Level** **getLevel** ()=0
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1709).*
- virtual void **setFormatter** (const **Formatter** *formatter)=0
*Sets the **Formatter** (p. 1699) used by this **Handler** (p. 1709).*
- virtual const **Formatter** * **getFormatter** ()=0
*Gets the **Formatter** (p. 1699) used by this **Handler** (p. 1709).*

6.325.1 Detailed Description

A **Handler** (p. 1709) object takes log messages from a **Logger** (p. 2028) and exports them. It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log, or whatever.

A **Handler** (p. 1709) can be disabled by doing a **setLevel**(**Level**.OFF) and can be re-enabled by doing a **setLevel** with an appropriate level.

Handler (p. 1709) classes typically use **LogManager** (p. 2045) properties to set default values for the Handler's **Filter** (p. 1630), **Formatter** (p. 1699), and **Level**. See the specific documentation for each concrete **Handler** (p. 1709) class.

6.325.2 Constructor & Destructor Documentation

6.325.2.1 `virtual decaf::util::logging::Handler::~~Handler ()` [inline, virtual]

6.325.3 Member Function Documentation

6.325.3.1 `virtual void decaf::util::logging::Handler::flush ()` [pure virtual]

Flush the Handler's output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 3078).

6.325.3.2 `virtual const Filter* decaf::util::logging::Handler::getFilter ()` [pure virtual]

Gets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.

Returns:

Filter (p. 1630) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 3078).

6.325.3.3 `virtual const Formatter* decaf::util::logging::Handler::getFormatter ()` [pure virtual]

Gets the **Formatter** (p. 1699) used by this **Handler** (p. 1709).

Returns:

Filter (p. 1630) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 3079).

6.325.3.4 `virtual Level decaf::util::logging::Handler::getLevel ()` [pure virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1709).

Returns:

Level enumeration value

Implemented in **decaf::util::logging::StreamHandler** (p. 3079).

6.325.3.5 `virtual void decaf::util::logging::Handler::isLoggable (const LogRecord & record)` [pure virtual]

Check if this **Handler** (p. 1709) would actually log a given **LogRecord** (p. 2050). This method checks if the **LogRecord** (p. 2050) has an appropriate Level and whether it satisfies any **Filter** (p. 1630). It also may make other **Handler** (p. 1709) specific checks that might prevent a handler from logging the **LogRecord** (p. 2050).

Parameters:

record LogRecord (p. 2050) to check

Implemented in **decaf::util::logging::StreamHandler** (p. 3079).

6.325.3.6 virtual void decaf::util::logging::Handler::publish (const LogRecord & *record*) [pure virtual]

Publish the Log Record to this **Handler** (p. 1709).

Parameters:

record The Log Record to Publish

Implemented in **decaf::util::logging::StreamHandler** (p. 3079).

6.325.3.7 virtual void decaf::util::logging::Handler::setFilter (const Filter * *filter*) [pure virtual]

Sets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records. For each call of publish the **Handler** (p. 1709) will call this **Filter** (p. 1630) (if it is non-null) to check if the **LogRecord** (p. 2050) should be published or discarded.

Parameters:

filter Filter (p. 1630) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 3080).

6.325.3.8 virtual void decaf::util::logging::Handler::setFormatter (const Formatter * *formatter*) [pure virtual]

Sets the **Formatter** (p. 1699) used by this **Handler** (p. 1709). Some Handlers may not use Formatters, in which case the **Formatter** (p. 1699) will be remembered, but not used.

Parameters:

formatter Filter (p. 1630) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 3080).

6.325.3.9 virtual void decaf::util::logging::Handler::setLevel (Level *value*) [pure virtual]

Set (p. 2905) the log level specifying which message levels will be logged by this **Handler** (p. 1709). The intention is to allow developers to turn on voluminous logging, but to limit the messages that are sent to certain Handlers.

Parameters:

value Level enumeration value

Implemented in **decaf::util::logging::StreamHandler** (p. 3080).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Handler.h`

6.326 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)
Create a new HexParser.
- virtual **~HexStringParser** ()
- long long **parse** (const std::string &hexString)
Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

6.326.1 Constructor & Destructor Documentation

6.326.1.1 decaf::internal::util::HexStringParser::HexStringParser (int exponentWidth, int mantissaWidth)

Create a new HexParser.

Parameters:

exponentWidth - Width of the exponent for the type to parse
mantissaWidth - Width of the mantissa for the type to parse

6.326.1.2 virtual decaf::internal::util::HexStringParser::~~HexStringParser () [inline, virtual]

6.326.2 Member Function Documentation

6.326.2.1 long long decaf::internal::util::HexStringParser::parse (const std::string &hexString)

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Parameters:

hexString - string to parse

Returns:

the bits parsed from the string

6.326.2.2 static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

6.326.2.3 static float decaf::internal::util::HexStringParser::parseFloat (const std::string & *hexString*) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**HexStringParser.h**

6.327 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p.1715) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>
```

Public Member Functions

- **HexTable** ()
- virtual **~HexTable** ()
- virtual const std::string & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

- virtual const std::string & **operator[]** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **size** () const

Returns the max size of this Table.

6.327.1 Detailed Description

The **HexTable** (p.1715) class maps hexadecimal strings to the value of an index into the table, i.e. the class will return "FF" for the index 255 in the table.

6.327.2 Constructor & Destructor Documentation

6.327.2.1 activemq::wireformat::openwire::utils::HexTable::HexTable ()

6.327.2.2 virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable ()
[inline, virtual]

6.327.3 Member Function Documentation

6.327.3.1 virtual const std::string&
activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t
index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException
) [virtual]

6.327.3.2 virtual const std::string&
activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t
index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
[virtual]

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

Parameters:

index The index of the value in the table to fetch.

Returns:

string containing the hex value if the index

Exceptions:

IndexOutOfBoundsException

6.327.3.3 `virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size
() const [inline, virtual]`

Returns the max size of this Table.

Returns:

an integer size value

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/HexTable.h`

6.328 decaf::net::HttpRetryException Class Reference

#include <src/main/decaf/net/HttpRetryException.h> Inheritance diagram for decaf::net::HttpRetryException:

Public Member Functions

- **HttpRetryException** () throw ()
Default Constructor.
- **HttpRetryException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()
Copy Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **HttpRetryException** (const std::exception *cause) throw ()
Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **HttpRetryException** * **clone** () const
Clones this exception.
- virtual ~**HttpRetryException** () throw ()

6.328.1 Constructor & Destructor Documentation

6.328.1.1 decaf::net::HttpRetryException::HttpRetryException () throw () [inline]

Default Constructor.

6.328.1.2 decaf::net::HttpRetryException::HttpRetryException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.328.1.3 decaf::net::HttpRetryException::HttpRetryException (const HttpRetryException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.328.1.4 decaf::net::HttpRetryException::HttpRetryException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.328.1.5 decaf::net::HttpRetryException::HttpRetryException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.328.1.6 decaf::net::HttpRetryException::HttpRetryException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.328.1.7 virtual decaf::net::HttpRetryException::~~HttpRetryException () throw
() [inline, virtual]

6.328.2 Member Function Documentation

6.328.2.1 virtual HttpRetryException* decaf::net::HttpRetryException::clone ()
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**HttpRetryException.h**

6.329 decaf::lang::exceptions::IllegalArgumentException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h> Inheritance diagram for decaf::lang::exceptions::IllegalArgumentException:

Public Member Functions

- **IllegalArgumentException** () throw ()
Default Constructor.
- **IllegalArgumentException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()
Copy Constructor.
- **IllegalArgumentException** (const std::exception *cause) throw ()
Constructor.
- **IllegalArgumentException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalArgumentException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **IllegalArgumentException** * clone () const
Clones this exception.
- virtual ~**IllegalArgumentException** () throw ()

6.329.1 Constructor & Destructor Documentation

6.329.1.1 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException () throw () [inline]

Default Constructor.

6.329.1.2 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.329.1.3 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const IllegalArgumentException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.329.1.4 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.329.1.5 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.329.1.6 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.329.1.7 **virtual**
decaf::lang::exceptions::IllegalArgumentException::~~IllegalArgumentException
() throw () [inline, virtual]

6.329.2 **Member Function Documentation**

6.329.2.1 **virtual IllegalArgumentException* de-**
caf::lang::exceptions::IllegalArgumentException::clone ()
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalArgumentException.h`

6.330 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h> Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

Public Member Functions

- **IllegalMonitorStateException** () throw ()
Default Constructor.
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()
Copy Constructor.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalMonitorStateException** * clone () const
Clones this exception.
- virtual ~**IllegalMonitorStateException** () throw ()

6.330.1 Constructor & Destructor Documentation

6.330.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException () throw () [inline]

Default Constructor.

6.330.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.330.1.3 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const IllegalMonitorStateException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.330.1.4 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.330.1.5 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.330.1.6 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.330.1.7 **virtual**
 decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException
 () throw () [inline, virtual]

6.330.2 Member Function Documentation

6.330.2.1 **virtual IllegalMonitorStateException* de-**
 caf::lang::exceptions::IllegalMonitorStateException::clone
 () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalMonitorStateException.h**

6.331 decaf::lang::exceptions::IllegalStateException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalStateException.h> Inheritance diagram for decaf::lang::exceptions::IllegalStateException:

Public Member Functions

- **IllegalStateException** () throw ()
Default Constructor.
- **IllegalStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
Copy Constructor.
- **IllegalStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalStateException** * clone () const
Clones this exception.
- virtual ~**IllegalStateException** () throw ()

6.331.1 Constructor & Destructor Documentation

6.331.1.1 decaf::lang::exceptions::IllegalStateException::IllegalStateException () throw () [inline]

Default Constructor.

6.331.1.2 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.331.1.3 decaf::lang::exceptions::IllegalStateException::IllegalStateException
(const IllegalStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.331.1.4 decaf::lang::exceptions::IllegalStateException::IllegalStateException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.331.1.5 decaf::lang::exceptions::IllegalStateException::IllegalStateException
(const char * *file*, const int *lineNumber*, const std::exception * *cause*,
const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.331.1.6 decaf::lang::exceptions::IllegalStateException::IllegalStateException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.331.1.7 **virtual**
decaf::lang::exceptions::IllegalStateException::~~IllegalStateException ()
throw () [inline, virtual]

6.331.2 Member Function Documentation

6.331.2.1 **virtual** **IllegalStateException*** **de-**
caf::lang::exceptions::IllegalStateException::clone () **const**
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 1815).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalStateException.h`

6.332 cms::IllegalStateException Class Reference

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

#include <src/main/cms/IllegalStateException.h> Inheritance diagram for cms::IllegalStateException:

Public Member Functions

- **IllegalStateException** () throw ()
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**IllegalStateException** () throw ()

6.332.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation. For example, this exception must be thrown if **Session.commit** (p. 2843) is called on a non-transacted session.

Since:

1.3

6.332.2 Constructor & Destructor Documentation

6.332.2.1 cms::IllegalStateException::IllegalStateException () throw ()

6.332.2.2 cms::IllegalStateException::IllegalStateException (const **IllegalStateException** & *ex*) throw ()

6.332.2.3 cms::IllegalStateException::IllegalStateException (const std::string & *message*, const std::exception * *cause*) throw ()

6.332.2.4 cms::IllegalStateException::IllegalStateException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.332.2.5 virtual cms::IllegalStateException::~~IllegalStateException () throw ()
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**IllegalStateException.h**

6.333 decaf::lang::exceptions::IllegalThreadStateException Class Reference

#include <src/main/decaf/lang/exceptions/IllegalThreadStateException.h> Inheritance diagram for decaf::lang::exceptions::IllegalThreadStateException:

Public Member Functions

- **IllegalThreadStateException** () throw ()
Default Constructor.
- **IllegalThreadStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **IllegalThreadStateException** (const **IllegalThreadStateException** &ex) throw ()
Copy Constructor.
- **IllegalThreadStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalThreadStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalThreadStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalThreadStateException** * clone () const
Clones this exception.
- virtual ~**IllegalThreadStateException** () throw ()

6.333.1 Constructor & Destructor Documentation

6.333.1.1 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException () throw () [inline]

Default Constructor.

6.333.1.2 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.333.1.3 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const IllegalThreadStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.333.1.4 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.333.1.5 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.333.1.6 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.333.1.7 **virtual**
 decaf::lang::exceptions::IllegalThreadStateException::~~IllegalThreadStateException
 () throw () [inline, virtual]

6.333.2 **Member Function Documentation**

6.333.2.1 **virtual IllegalThreadStateException* de-**
 caf::lang::exceptions::IllegalThreadStateException::clone ()
 const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p.1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalThreadStateException.h`

6.334 activemq::transport::inactivity::InactivityMonitor Class Reference

#include <src/main/activemq/transport/inactivity/InactivityMonitor.h> Inheritance diagram for activemq::transport::inactivity::InactivityMonitor:

Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual ~**InactivityMonitor** ()
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

6.334.1 Constructor & Destructor Documentation

- 6.334.1.1 `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const Pointer< wireformat::WireFormat > & wireFormat)`
- 6.334.1.2 `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const decaf::util::Properties & properties, const Pointer< wireformat::WireFormat > & wireFormat)`
- 6.334.1.3 `virtual
activemq::transport::inactivity::InactivityMonitor::~~InactivityMonitor ()
[virtual]`

6.334.2 Member Function Documentation

- 6.334.2.1 `virtual void activemq::transport::inactivity::InactivityMonitor::close ()
throw (decaf::io::IOException) [virtual]`

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

IOException if an error occurs while closing the **Transport** (p. 3273).

Reimplemented from **activemq::transport::TransportFilter** (p. 3283).

- 6.334.2.2 `long long ac-
tivemq::transport::inactivity::InactivityMonitor::getInitialDelayTime ()
const`
- 6.334.2.3 `long long ac-
tivemq::transport::inactivity::InactivityMonitor::getReadCheckTime ()
const`
- 6.334.2.4 `long long ac-
tivemq::transport::inactivity::InactivityMonitor::getWriteCheckTime ()
const`
- 6.334.2.5 `bool ac-
tivemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired
() const`
- 6.334.2.6 `virtual void ac-
tivemq::transport::inactivity::InactivityMonitor::onCommand (const
Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters:

command - the received command object.

Reimplemented from `activemq::transport::TransportFilter` (p. 3285).

6.334.2.7 `virtual void activemq::transport::inactivity::InactivityMonitor::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3285).

6.334.2.8 `virtual void activemq::transport::inactivity::InactivityMonitor::onException (const decaf::lang::Exception & ex)` [virtual]

Event handler for an exception from a command transport.

Parameters:

ex The exception to handle.

Reimplemented from `activemq::transport::TransportFilter` (p. 3286).

- 6.334.2.9 `void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime`
 (`long long value`) `const`
- 6.334.2.10 `void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired`
 (`bool value`)
- 6.334.2.11 `void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime`
 (`long long value`)
- 6.334.2.12 `void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime`
 (`long long value`)

6.334.3 Friends And Related Function Documentation

- 6.334.3.1 `friend class AsyncSignalReadErrorTask` `[friend]`
- 6.334.3.2 `friend class AsyncWriteTask` `[friend]`
- 6.334.3.3 `friend class ReadChecker` `[friend]`
- 6.334.3.4 `friend class WriteChecker` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/InactivityMonitor.h`

6.335 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>Inheritance diagram for decaf::lang::exceptions::IndexOutOfBoundsException:

Public Member Functions

- **IndexOutOfBoundsException** () throw ()
Default Constructor.
- **IndexOutOfBoundsException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()
Copy Constructor.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const std::exception *cause) throw ()
Constructor.
- virtual **IndexOutOfBoundsException** * clone () const
Clones this exception.
- virtual ~**IndexOutOfBoundsException** () throw ()

6.335.1 Constructor & Destructor Documentation

6.335.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException () throw () [inline]

Default Constructor.

6.335.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.335.1.3 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const IndexOutOfBoundsException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.335.1.4 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.335.1.5 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.335.1.6 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.335.1.7 virtual
decaf::lang::exceptions::IndexOutOfBoundsException::~IndexOutOfBoundsException
() throw () [inline, virtual]

6.335.2 Member Function Documentation

6.335.2.1 virtual IndexOutOfBoundsException* decaf::lang::exceptions::IndexOutOfBoundsException::clone
() const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IndexOutOfBoundsException.h**

6.336 decaf::io::InputStream Class Reference

Base interface for an input stream.

```
#include <src/main/decaf/io/InputStream.h>
Inheritance diagram for decaf::io::InputStream:
```

Public Member Functions

- virtual **~InputStream** ()
- virtual void **mark** (int readLimit)=0

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** ()=0 throw (IOException)

Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const =0

Determines if this input stream supports the mark and reset methods.
- virtual std::size_t **available** () const =0 throw (IOException)

Indicates the number of bytes available.
- virtual int **read** ()=0 throw (IOException)

Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize)=0 throw (IOException, lang::exceptions::NullPointerException)

Reads an array of bytes from the buffer.
- virtual std::size_t **skip** (std::size_t num)=0 throw (io::IOException, lang::exceptions::UnsupportedOperationException)

Skips over and discards n bytes of data from this input stream.

6.336.1 Detailed Description

Base interface for an input stream.

6.336.2 Constructor & Destructor Documentation

6.336.2.1 virtual decaf::io::InputStream::~~InputStream () [inline, virtual]

6.336.3 Member Function Documentation

6.336.3.1 virtual std::size_t decaf::io::InputStream::available () const throw (IOException) [pure virtual]

Indicates the number of bytes available.

Returns:

the number of bytes available on this input stream.

Exceptions:

IOException (p. 1820) if an error occurs.

Implemented in `decaf::internal::io::StandardInputStream` (p. 3001), `decaf::io::BlockingByteArrayInputStream` (p. 726), `decaf::io::BufferedInputStream` (p. 810), `decaf::io::ByteArrayInputStream` (p. 894), `decaf::io::FilterInputStream` (p. 1633), and `decaf::net::SocketInputStream` (p. 2979).

6.336.3.2 virtual void decaf::io::InputStream::mark (int *readLimit*) [pure virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implemented in `decaf::internal::io::StandardInputStream` (p. 3002), `decaf::io::BlockingByteArrayInputStream` (p. 727), `decaf::io::BufferedInputStream` (p. 811), `decaf::io::ByteArrayInputStream` (p. 895), `decaf::io::FilterInputStream` (p. 1634), and `decaf::net::SocketInputStream` (p. 2979).

6.336.3.3 virtual bool decaf::io::InputStream::markSupported () const [pure virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implemented in `decaf::internal::io::StandardInputStream` (p. 3002), `decaf::io::BlockingByteArrayInputStream` (p. 727), `decaf::io::BufferedInputStream` (p. 811), `decaf::io::ByteArrayInputStream` (p. 895), `decaf::io::FilterInputStream` (p. 1634), and `decaf::net::SocketInputStream` (p. 2979).

6.336.3.4 `virtual int decaf::io::InputStream::read (unsigned char * buffer,
std::size_t offset, std::size_t bufferSize) throw (IOException,
lang::exceptions::NullPointerException) [pure virtual]`

Reads an array of bytes from the buffer. Blocks until the requested number of bytes are available.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF is detected

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException if buffer is null

Implemented in `activemq::io::LoggingInputStream` (p. 2038), `decaf::internal::io::StandardInputStream` (p. 3003), `decaf::io::BlockingByteArrayInputStream` (p. 728), `decaf::io::BufferedInputStream` (p. 811), `decaf::io::ByteArrayInputStream` (p. 896), `decaf::io::DataInputStream` (p. 1381), `decaf::io::FilterInputStream` (p. 1635), and `decaf::net::SocketInputStream` (p. 2980).

6.336.3.5 `virtual int decaf::io::InputStream::read () throw (IOException) [pure virtual]`

Reads a single byte from the buffer. The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

Returns:

The next byte or -1 if the end of stream is reached.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implemented in `activemq::io::LoggingInputStream` (p. 2039), `decaf::internal::io::StandardInputStream` (p. 3004), `decaf::io::BlockingByteArrayInputStream` (p. 728), `decaf::io::BufferedInputStream` (p. 812), `decaf::io::ByteArrayInputStream` (p. 896), `decaf::io::DataInputStream` (p. 1382), `decaf::io::FilterInputStream` (p. 1636), and `decaf::net::SocketInputStream` (p. 2981).

6.336.3.6 `virtual void decaf::io::InputStream::reset () throw (IOException) [pure virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: * If the method mark has not been

called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1820) might be thrown. * If such an **IOException** (p. 1820) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1820). * If an **IOException** (p. 1820) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException (p. 1820)

Implemented in **decaf::internal::io::StandardInputStream** (p. 3004), **decaf::io::BlockingByteArrayInputStream** (p. 728), **decaf::io::BufferedInputStream** (p. 812), **decaf::io::ByteArrayInputStream** (p. 897), **decaf::io::FilterInputStream** (p. 1636), and **decaf::net::SocketInputStream** (p. 2981).

6.336.3.7 virtual std::size_t decaf::io::InputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1740) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException (p. 1820) if an error occurs

Implemented in **decaf::internal::io::StandardInputStream** (p. 3004), **decaf::io::BlockingByteArrayInputStream** (p. 729), **decaf::io::BufferedInputStream** (p. 812), **decaf::io::ByteArrayInputStream** (p. 897), **decaf::io::DataInputStream** (p. 1387), **decaf::io::FilterInputStream** (p. 1637), and **decaf::net::SocketInputStream** (p. 2981).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InputStream.h**

6.337 decaf::internal::nio::IntArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/IntArrayBuffer.h> Inheritance diagram for decaf::internal::nio::IntArrayBuffer:

Public Member Functions

- **IntArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1744) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **IntArrayBuffer** (int *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **IntArrayBuffer** (p. 1744) object that wraps the given array.*
- **IntArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset.*
- **IntArrayBuffer** (const IntArrayBuffer &other)
*Create a **IntArrayBuffer** (p. 1744) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~**IntArrayBuffer** ()
- virtual int * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the int array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual IntBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only int buffer that shares this buffer's content.
- virtual IntBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual IntBuffer * **duplicate** ()
Creates a new int buffer that shares this buffer's content.
- virtual int **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual int **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible int array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual IntBuffer & **put** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual IntBuffer & **put** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual IntBuffer * **slice** () const
Creates a new IntBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this *ByteArrayBuffer* (p. 870) as Read-Only.*

6.337.1 Constructor & Destructor Documentation

6.337.1.1 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **IntArrayBuffer** (p. 1744) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.
readOnly - should this buffer be read-only, default as false

6.337.1.2 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **IntArrayBuffer** (p. 1744) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.337.1.3 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p.908) and start at the given offset. The capacity and limit of the new **IntArrayBuffer** (p.1744) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p.908) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.337.1.4 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const IntArrayBuffer & other)`

Create a **IntArrayBuffer** (p.1744) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p.908) and when changes are made to that data it is reflected in both.

Parameters:

other - the **IntArrayBuffer** (p.1744) this one is to mirror.

6.337.1.5 `virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer () [virtual]`

6.337.2 Member Function Documentation

6.337.2.1 `virtual int* decaf::internal::nio::IntArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the int array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1753).

6.337.2.2 `virtual std::size_t decaf::internal::nio::IntArrayBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1754).

6.337.2.3 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer
() const [virtual]`

Creates a new, read-only int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 1754).

6.337.2.4 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact () throw
(decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 807) is copied

to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index **limit()** (p. 807) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this `IntBuffer`

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1754).

6.337.2.5 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate ()`
[virtual]

Creates a new int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new int Buffer which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1755).

6.337.2.6 `virtual int decaf::internal::nio::IntArrayBuffer::get (std::size_t index)`
`const throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the int is to be read

Returns:

the int that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::IntBuffer** (p. 1756).

6.337.2.7 `virtual int decaf::internal::nio::IntArrayBuffer::get () throw (`
`decaf::nio::BufferUnderflowException)` [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the int at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::IntBuffer** (p. 1757).

6.337.2.8 virtual bool decaf::internal::nio::IntArrayBuffer::hasArray () const
[inline, virtual]

Tells whether or not this buffer is backed by an accessible int array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::IntBuffer** (p. 1757).

6.337.2.9 virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly () const
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 806).

6.337.2.10 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1758).

6.337.2.11 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::IntBuffer` (p. 1758).

6.337.2.12 `virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this `ByteBuffer` (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.337.2.13 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice () const [virtual]`

Creates a new `IntBuffer` whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create `IntBuffer` which the caller owns.

Implements `decaf::nio::IntBuffer` (p. 1760).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/IntArrayBuffer.h`

6.338 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:.

#include <src/main/decaf/nio/IntBuffer.h>Inheritance diagram for decaf::nio::IntBuffer:

Public Member Functions

- virtual **~IntBuffer** ()
- virtual std::string **toString** () const
- virtual int * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the int array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **IntBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only int buffer that shares this buffer's content.
- virtual **IntBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **IntBuffer** * **duplicate** ()=0
Creates a new int buffer that shares this buffer's content.
- virtual int **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual int **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **IntBuffer** & **get** (std::vector< int > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **IntBuffer** & **get** (int *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible int array.
- **IntBuffer** & **put** (**IntBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the ints remaining in the given source buffer into this buffer.

- **IntBuffer** & **put** (const int *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers ints into this buffer from the given source array.
- **IntBuffer** & **put** (std::vector< int > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source ints array into this buffer.
- virtual **IntBuffer** & **put** (int value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given ints into this buffer at the current position, and then increments the position.
- virtual **IntBuffer** & **put** (std::size_t index, int value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given ints into this buffer at the given index.
- virtual **IntBuffer** * **slice** () const =0
*Creates a new **IntBuffer** (p. 1751) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **IntBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **IntBuffer** &value) const
- virtual bool **operator==** (const **IntBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **IntBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **IntBuffer** * **allocate** (std::size_t capacity)
Allocates a new Double buffer.
- static **IntBuffer** * **wrap** (int *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **IntBuffer** (p. 1751).*
- static **IntBuffer** * **wrap** (std::vector< int > &buffer)
*Wraps the passed STL int Vector in a **IntBuffer** (p. 1751).*

Protected Member Functions

- **IntBuffer** (std::size_t capacity)
*Creates a **IntBuffer** (p. 1751) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.338.1 Detailed Description

This class defines four categories of operations upon int buffers: o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.338.2 Constructor & Destructor Documentation

6.338.2.1 decaf::nio::IntBuffer::IntBuffer (std::size_t *capacity*) [protected]

Creates a **IntBuffer** (p.1751) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p.803) in doubles

6.338.2.2 virtual decaf::nio::IntBuffer::~~IntBuffer () [inline, virtual]

6.338.3 Member Function Documentation

6.338.3.1 static IntBuffer* decaf::nio::IntBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in ints

Returns:

the **IntBuffer** (p.1751) that was allocated, caller owns.

6.338.3.2 virtual int* decaf::nio::IntBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the int array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 803)

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1746).

6.338.3.3 `virtual std::size_t decaf::nio::IntBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1747).

6.338.3.4 `virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1747).

6.338.3.5 `virtual IntBuffer& decaf::nio::IntBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]`

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 807) is copied

to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index `limit()` (p. 807) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **IntBuffer** (p. 1751)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1747).

6.338.3.6 `virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & value)
const [virtual]`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.338.3.7 `virtual IntBuffer* decaf::nio::IntBuffer::duplicate () [pure virtual]`

Creates a new int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new int **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1748).

6.338.3.8 `virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & value) const
[virtual]`

Returns:

true if this value is considered equal to the passed value.

6.338.3.9 `IntBuffer& decaf::nio::IntBuffer::get (int * buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)`

Relative bulk get method. This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies `length` ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters:

buffer - pointer to an allocated buffer to fill
offset - position in the buffer to start filling
length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` ints remaining in this buffer
NullPointerException if the passed buffer is null.

6.338.3.10 `IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > buffer) throw (BufferUnderflowException)`

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` ints remaining in this buffer

6.338.3.11 `virtual int decaf::nio::IntBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 803) where the int is to be read

Returns:

the int that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1748).

6.338.3.12 virtual int decaf::nio::IntBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the int at the current position

Exceptions:

BufferUnderflowException (p. 837) if there no more data to return

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1748).

6.338.3.13 virtual bool decaf::nio::IntBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible int array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1749).

6.338.3.14 virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & value) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.338.3.15 `virtual bool decaf::nio::IntBuffer::operator==(const IntBuffer & value) const` [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.338.3.16 `virtual IntBuffer& decaf::nio::IntBuffer::put (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given ints into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the ints to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1749).

6.338.3.17 `virtual IntBuffer& decaf::nio::IntBuffer::put (int value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given ints into this buffer at the current position, and then increments the position.

Parameters:

value - the ints value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1750).

6.338.3.18 IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source ints array into this buffer. This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters:

buffer - The buffer whose contents are copied to this **IntBuffer** (p. 1751)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.338.3.19 IntBuffer& decaf::nio::IntBuffer::put (const int * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers ints into this buffer from the given source array. If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 808), then no ints are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters:

buffer- The array from which ints are to be read

offset- The offset within the array of the first int to be read;

length - The number of ints to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.338.3.20 IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the ints remaining in the given source buffer into this buffer. If there are more ints remaining in the source buffer than in this buffer, that is, if `src.remaining() >`

remaining() (p. 808), then no ints are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `n = src.remaining()` ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters:

src - the buffer to take ints from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining ints in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.338.3.21 `virtual IntBuffer* decaf::nio::IntBuffer::slice () const` [pure virtual]

Creates a new **IntBuffer** (p. 1751) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **IntBuffer** (p. 1751) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1750).

6.338.3.22 `virtual std::string decaf::nio::IntBuffer::toString () const` [virtual]

Returns:

a `std::string` describing this object

6.338.3.23 `static IntBuffer* decaf::nio::IntBuffer::wrap (std::vector< int > & buffer)` [static]

Wraps the passed STL int Vector in a **IntBuffer** (p. 1751). The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new **IntBuffer** (p.1751) that is backed by *buffer*, caller owns.

6.338.3.24 `static IntBuffer* decaf::nio::IntBuffer::wrap (int *
array, std::size_t offset, std::size_t length) throw (
lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **IntBuffer** (p.1751). The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **IntBuffer** (p.1751) that is backed by *buffer*, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/IntBuffer.h`

6.339 decaf::lang::Integer Class Reference

#include <src/main/decaf/lang/Integer.h> Inheritance diagram for decaf::lang::Integer:

Public Member Functions

- **Integer** (int value)
- **Integer** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Integer** ()
- virtual int **compareTo** (const **Integer** &i) const
*Compares this **Integer** (p. 1762) instance with another.*
- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Integer** &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const int &i) const
*Compares this **Integer** (p. 1762) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const int &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Integer** **decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Integer** (p. 1762).*
- static int **reverseBytes** (int value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.
- static int **reverse** (int value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.
- static int **parseInt** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed int in the radix specified by the second argument.
- static int **parseInt** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal int.
- static **Integer** **valueOf** (int value)
*Returns a **Integer** (p. 1762) instance representing the specified int value.*
- static **Integer** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Integer** (p. 1762) object holding the value given by the specified std::string.*
- static **Integer** **valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Integer** (p. 1762) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*
- static int **bitCount** (int value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static std::string **toString** (int value)
Converts the int to a String representation.
- static std::string **toString** (int value, int radix)
Returns a string representation of the first argument in the radix specified by the second argument.
- static std::string **toHexString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 16.
- static std::string **toOctalString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 8.
- static std::string **toBinaryString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 2.

- static int **highestOneBit** (int value)
Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
- static int **lowestOneBit** (int value)
Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (int value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.
- static int **numberOfTrailingZeros** (int value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.
- static int **rotateLeft** (int value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.
- static int **rotateRight** (int value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.
- static int **signum** (int value)
Returns the signum function of the specified int value.

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const int **MAX_VALUE** = (int)0x7FFFFFFF
The maximum value that the primitive type can hold.
- static const int **MIN_VALUE** = (int)0x80000000
The minimum value that the primitive type can hold.

6.339.1 Constructor & Destructor Documentation

6.339.1.1 decaf::lang::Integer::Integer (int value)

Parameters:

value The primitive value to wrap in an Integer (p. 1762) instance.

6.339.1.2 decaf::lang::Integer::Integer (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value The base 10 encoded string to decode to an `Integer` (p.1762) and wrap.

Exceptions:

NumberFormatException

6.339.1.3 virtual decaf::lang::Integer::~~Integer () [inline, virtual]

6.339.2 Member Function Documentation

6.339.2.1 static int decaf::lang::Integer::bitCount (int *value*) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the population count.

Parameters:

value - the int to count

Returns:

the number of one-bits in the two's complement binary representation of the specified int value.

6.339.2.2 virtual unsigned char decaf::lang::Integer::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p.2432).

6.339.2.3 virtual int decaf::lang::Integer::compareTo (const int & *i*) const [virtual]

Compares this `Integer` (p.1762) instance with another.

Parameters:

i - the `Integer` (p.1762) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< int >` (p.1083).

6.339.2.4 `virtual int decaf::lang::Integer::compareTo (const Integer & i) const`
[virtual]

Compares this **Integer** (p.1762) instance with another.

Parameters:

i - the **Integer** (p.1762) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Integer** > (p.1083).

6.339.2.5 `static Integer decaf::lang::Integer::decode (const std::string & value)`
`throw (exceptions::NumberFormatException)` [static]

Decodes a String into a **Integer** (p.1762). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Integer** (p.1762) object containing the decoded value

Exceptions:

NumberFormatException if the string is not formatted correctly.

6.339.2.6 `virtual double decaf::lang::Integer::doubleValue () const` [inline,
virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p.2433).

6.339.2.7 `bool decaf::lang::Integer::equals (const int & i) const [inline, virtual]`**Parameters:**

i - the **Integer** (p.1762) object to compare against.

Returns:

true if the two **Integer** (p.1762) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p.1084).

6.339.2.8 `bool decaf::lang::Integer::equals (const Integer & i) const [inline, virtual]`**Parameters:**

i - the **Integer** (p.1762) object to compare against.

Returns:

true if the two **Integer** (p.1762) Objects have the same value.

Implements **decaf::lang::Comparable**< **Integer** > (p.1084).

6.339.2.9 `virtual float decaf::lang::Integer::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p.2433).

6.339.2.10 `static int decaf::lang::Integer::highestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.339.2.11 `virtual int decaf::lang::Integer::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.339.2.12 `virtual long long decaf::lang::Integer::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.339.2.13 `static int decaf::lang::Integer::lowestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.339.2.14 `static int decaf::lang::Integer::numberOfLeadingZeros (int value) [static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

$\text{* floor}(\log_2(x)) = 31 - \text{numberOfLeadingZeros}(x)$ * $\text{* ceil}(\log_2(x)) = 32 - \text{numberOfLeadingZeros}(x - 1)$

Parameters:

value - the int to be inspected

Returns:

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.339.2.15 **static int decaf::lang::Integer::numberOfTrailingZeros (int *value*)**
 [static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.339.2.16 **virtual bool decaf::lang::Integer::operator< (const int & *i*) const**
 [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 1084).

6.339.2.17 **virtual bool decaf::lang::Integer::operator< (const Integer & *i*) const**
 [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Integer >** (p. 1084).

6.339.2.18 **virtual bool decaf::lang::Integer::operator== (const int & *i*) const**
 [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 1085).

6.339.2.19 `virtual bool decaf::lang::Integer::operator==(const Integer & i) const`
`[inline, virtual]`

Compares equality between this object and the one passed.

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Integer >** (p. 1085).

6.339.2.20 `static int decaf::lang::Integer::parseInt (const std::string & s) throw (`
`exceptions::NumberFormatException) [static]`

Parses the string argument as a signed decimal int. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseInt(const std::string, int)` method.

Parameters:

s - String to convert to a int

Returns:

the converted int value

Exceptions:

NumberFormatException if the string is not a int.

6.339.2.21 `static int decaf::lang::Integer::parseInt (const std::string & s, int radix)`
`throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed int in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 985) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:
 * The first argument is null or is a string of length zero. * The radix is either smaller than

Character.MIN_RADIX (p. 989) or larger than **Character.MAX_RADIX** (p. 989). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type int.

Parameters:

s - the String containing the int representation to be parsed
radix - the radix to be used while parsing s

Returns:

the int represented by the string argument in the specified radix.

Exceptions:

NumberFormatException - If String does not contain a parsable int.

6.339.2.22 static int decaf::lang::Integer::reverse (int *value*) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

Parameters:

value - the value whose bits are to be reversed

Returns:

the reversed bits int.

6.339.2.23 static int decaf::lang::Integer::reverseBytes (int *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

Parameters:

value - the int whose bytes we are to reverse

Returns:

the reversed int.

6.339.2.24 static int decaf::lang::Integer::rotateLeft (int *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters:

value - the int to be inspected
distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

6.339.2.25 `static int decaf::lang::Integer::rotateRight (int value, int distance)`
 `[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

Parameters:

value - the int to be inspected
distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

6.339.2.26 `virtual short decaf::lang::Integer::shortValue () const` `[inline, virtual]`

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2434).

6.339.2.27 `static int decaf::lang::Integer::signum (int value)` `[static]`

Returns the signum function of the specified int value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters:

value - the int to be inspected

Returns:

the signum function of the specified int value.

6.339.2.28 `static std::string decaf::lang::Integer::toBinaryString (int value)`
 [static]

Returns a string representation of the integer argument as an unsigned integer in base 2. The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

Parameters:

value - the int to be translated to a binary string

Returns:

the unsigned int value as a binary string

6.339.2.29 `static std::string decaf::lang::Integer::toHexString (int value)` [static]

Returns a string representation of the integer argument as an unsigned integer in base 16. The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters:

value - the int to be translated to an Octal string

Returns:

the unsigned int value as a Octal string

6.339.2.30 `static std::string decaf::lang::Integer::toOctalString (int value)` [static]

Returns a string representation of the integer argument as an unsigned integer in base 8. The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters:

value - the int to be translated to an Octal string

Returns:

the unsigned int value as a Octal string

6.339.2.31 `static std::string decaf::lang::Integer::toString (int value, int radix)`
[static]

Returns a string representation of the first argument in the radix specified by the second argument. If the radix is smaller than **Character.MIN_RADIX** (p.989) or larger than **Character.MAX_RADIX** (p.989), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

Parameters:

value - the int to convert to a string

radix - the radix to format the string in

Returns:

an int formatted to the string value of the radix given.

6.339.2.32 `static std::string decaf::lang::Integer::toString (int value)` [static]

Converts the int to a String representation.

Parameters:

value The int to convert to a `std::string` instance.

Returns:

string representation

6.339.2.33 `std::string decaf::lang::Integer::toString () const`

Returns:

this Integer (p.1762) Object as a String Representation

6.339.2.34 `static Integer decaf::lang::Integer::valueOf (const std::string & value, int radix) throw (exceptions::NumberFormatException)` [static]

Returns a **Integer** (p.1762) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as

representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the `parseInt(std::string, int)` method. The result is a **Integer** (p. 1762) object that represents the int value specified by the string.

Parameters:

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Integer** (p. 1762) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid int.

6.339.2.35 `static Integer decaf::lang::Integer::valueOf (const std::string & value)
 throw (exceptions::NumberFormatException) [static]`

Returns a **Integer** (p. 1762) object holding the value given by the specified std::string. The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the `parseInt(std::string)` method. The result is a **Integer** (p. 1762) object that represents the int value specified by the string.

Parameters:

value - std::string to parse as base 10

Returns:

new **Integer** (p. 1762) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal int.

6.339.2.36 `static Integer decaf::lang::Integer::valueOf (int value) [inline, static]`

Returns a **Integer** (p. 1762) instance representing the specified int value.

Parameters:

value - the int to wrap

Returns:

the new **Integer** (p. 1762) object wrapping value.

6.339.3 Field Documentation

6.339.3.1 `const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF
 [static]`

The maximum value that the primitive type can hold.

6.339.3.2 `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000` [static]

The minimum value that the primitive type can hold.

6.339.3.3 `const int decaf::lang::Integer::SIZE = 32` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Integer.h`

6.340 activemq::commands::IntegerResponse Class Reference

#include <src/main/activemq/commands/IntegerResponse.h> Inheritance diagram for activemq::commands::IntegerResponse:

Public Member Functions

- **IntegerResponse** ()
- virtual **~IntegerResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **IntegerResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual int **getResult** () const
- virtual void **setResult** (int result)

Static Public Attributes

- static const unsigned char **ID_INTEGERRESPONSE** = 34

Protected Member Functions

- **IntegerResponse** (const **IntegerResponse** &)
- **IntegerResponse & operator=** (const **IntegerResponse** &)

Protected Attributes

- int result

6.340.1 Constructor & Destructor Documentation

- 6.340.1.1** `activemq::commands::IntegerResponse::IntegerResponse (const IntegerResponse &) [inline, protected]`
- 6.340.1.2** `activemq::commands::IntegerResponse::IntegerResponse ()`
- 6.340.1.3** `virtual activemq::commands::IntegerResponse::~~IntegerResponse () [virtual]`

6.340.2 Member Function Documentation

- 6.340.2.1** `virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.2782).

- 6.340.2.2** `virtual void activemq::commands::IntegerResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::Response` (p.2782).

- 6.340.2.3** `virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p.1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p.2782).

- 6.340.2.4** `virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Reimplemented from **activemq::commands::Response** (p. 2783).

6.340.2.5 `virtual int activemq::commands::IntegerResponse::getResult () const`
[virtual]

6.340.2.6 `IntegerResponse& activemq::commands::IntegerResponse::operator=`
`(const IntegerResponse &) [inline, protected]`

6.340.2.7 `virtual void activemq::commands::IntegerResponse::setResult (int result)`
[virtual]

6.340.2.8 `virtual std::string activemq::commands::IntegerResponse::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2783).

6.340.3 Field Documentation

6.340.3.1 `const unsigned char activemq::commands::IntegerResponse::ID_ -`
`INTEGERRESPONSE = 34 [static]`

6.340.3.2 `int activemq::commands::IntegerResponse::result [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/IntegerResponse.h`

6.341 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1780).

#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.341.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1780). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.341.2 Constructor & Destructor Documentation

6.341.2.1 `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.341.2.2 `virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.341.3 Member Function Documentation

6.341.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.341.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.341.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2792).

6.341.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.341.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2793).

6.341.3.6 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2794).

6.341.3.7 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2795).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h

6.342 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1784).

#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.342.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1784). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.342.2 Constructor & Destructor Documentation

6.342.2.1 `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponseMarshaller()` `[inline]`

6.342.2.2 `virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` `[inline, virtual]`

6.342.3 Member Function Documentation

6.342.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.342.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.342.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2807).

6.342.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2808).

6.342.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2808).

6.342.3.6 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2809).

6.342.3.7 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2810).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h

6.343 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1788).

#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.343.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1788). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.343.2 Constructor & Destructor Documentation

6.343.2.1 `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponseMarshaller()` `[inline]`

6.343.2.2 `virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` `[inline, virtual]`

6.343.3 Member Function Documentation

6.343.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.343.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.343.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2797).

6.343.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.343.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2798).

6.343.3.6 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2799).

6.343.3.7 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2800).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h

6.344 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1792).

#include <src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.344.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1792). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.344.2 Constructor & Destructor Documentation

6.344.2.1 `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::IntegerResponseMarshaller()` [inline]

6.344.2.2 `virtual activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` [inline, virtual]

6.344.3 Member Function Documentation

6.344.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.344.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.344.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2812).

6.344.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.344.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2813).

6.344.3.6 virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2814).

6.344.3.7 virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2815).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h

6.345 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1796).

#include <src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.345.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1796). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.345.2 Constructor & Destructor Documentation

6.345.2.1 `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::IntegerResponseMarshaller()` `[inline]`

6.345.2.2 `virtual activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::~~IntegerResponseMarshaller()` `[inline, virtual]`

6.345.3 Member Function Documentation

6.345.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.345.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.345.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2802).

6.345.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803).

6.345.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2803).

6.345.3.6 virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2804).

6.345.3.7 virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h

6.346 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 2371).

#include <src/main/activemq/transport/mock/InternalCommandListener.h>Inheritance diagram for activemq::transport::mock::InternalCommandListener:

Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** *transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- void **run** ()

6.346.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 2371). This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 2785) and getting a set of Commands to send back into the **MockTransport** (p. 2371) as incoming Commands and Responses.

6.346.2 Constructor & Destructor Documentation

6.346.2.1 **activemq::transport::mock::InternalCommandListener::InternalCommandListener** ()

6.346.2.2 **virtual**
activemq::transport::mock::InternalCommandListener::~~InternalCommandListener
 () [virtual]

6.346.3 Member Function Documentation

6.346.3.1 **virtual void** **activemq::transport::mock::InternalCommandListener::onCommand** (const **Pointer**< **Command** > & *command*) [virtual]

Event handler for the receipt of a command. The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3273) deletes the command upon receipt.

Parameters:

command the received command object.

Implements **activemq::transport::TransportListener** (p. 3289).

6.346.3.2 void activemq::transport::mock::InternalCommandListener::run ()

6.346.3.3 void activemq::transport::mock::InternalCommandListener::setResponseBuilder (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

6.346.3.4 void activemq::transport::mock::InternalCommandListener::setTransport (MockTransport * *transport*) [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**InternalCommandListener.h**

6.347 decaf::lang::exceptions::InterruptedException Class Reference

#include <src/main/decaf/lang/exceptions/InterruptedException.h> Inheritance diagram for decaf::lang::exceptions::InterruptedException:

Public Member Functions

- **InterruptedException** () throw ()
Default Constructor.
- **InterruptedException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **InterruptedException** (const **InterruptedException** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception *cause) throw ()
Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InterruptedException** * **clone** () const
Clones this exception.
- virtual ~**InterruptedException** () throw ()

6.347.1 Constructor & Destructor Documentation

6.347.1.1 decaf::lang::exceptions::InterruptedException::InterruptedException () throw () [inline]

Default Constructor.

6.347.1.2 decaf::lang::exceptions::InterruptedException::InterruptedException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.347.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException
(const InterruptedException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.347.1.4 decaf::lang::exceptions::InterruptedException::InterruptedException
(const char * *file*, const int *lineNumber*, const std::exception * *cause*,
const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.347.1.5 decaf::lang::exceptions::InterruptedException::InterruptedException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.347.1.6 decaf::lang::exceptions::InterruptedException::InterruptedException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.347.1.7 **virtual**
 decaf::lang::exceptions::InterruptedException::~~InterruptedException ()
 throw () [inline, virtual]

6.347.2 Member Function Documentation

6.347.2.1 **virtual InterruptedException* de-**
 caf::lang::exceptions::InterruptedException::clone () const
 [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InterruptedException.h**

6.348 decaf::io::InterruptedIOException Class Reference

#include <src/main/decaf/io/InterruptedIOException.h> Inheritance diagram for decaf::io::InterruptedIOException:

Public Member Functions

- **InterruptedIOException** () throw ()
Default Constructor.
- **InterruptedIOException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **InterruptedIOException** (const InterruptedIOException &ex) throw ()
Copy Constructor.
- **InterruptedIOException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedIOException** (const std::exception *cause) throw ()
Constructor.
- **InterruptedIOException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **InterruptedIOException** * clone () const
Clones this exception.
- virtual ~**InterruptedIOException** () throw ()

6.348.1 Constructor & Destructor Documentation

6.348.1.1 decaf::io::InterruptedIOException::InterruptedIOException () throw () [inline]

Default Constructor.

6.348.1.2 decaf::io::InterruptedIOException::InterruptedIOException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.348.1.3 `decaf::io::InterruptedIOException::InterruptedIOException (const InterruptedIOException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.348.1.4 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.348.1.5 `decaf::io::InterruptedIOException::InterruptedIOException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.348.1.6 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.348.1.7 virtual decaf::io::InterruptedIOException::~~InterruptedIOException ()
throw () [inline, virtual]

6.348.2 Member Function Documentation

6.348.2.1 virtual InterruptedIOException* decaf::io::InterruptedIOException::clone
() const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1822).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 2992).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

6.349 cms::InvalidClientIdException Class Reference

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

#include <src/main/cms/InvalidClientIdException.h> Inheritance diagram for cms::InvalidClientIdException:

Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

6.349.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Since:

1.3

6.349.2 Constructor & Destructor Documentation

- 6.349.2.1** cms::InvalidClientIdException::InvalidClientIdException () throw ()
- 6.349.2.2** cms::InvalidClientIdException::InvalidClientIdException (const InvalidClientIdException & ex) throw ()
- 6.349.2.3** cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception * cause) throw ()
- 6.349.2.4** cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()
- 6.349.2.5** virtual cms::InvalidClientIdException::~InvalidClientIdException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidClientIdException.h**

6.350 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

#include <src/main/cms/InvalidDestinationException.h> Inheritance diagram for cms::InvalidDestinationException:

Public Member Functions

- **InvalidDestinationException** () throw ()
- **InvalidDestinationException** (const **InvalidDestinationException** &ex) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidDestinationException** () throw ()

6.350.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Since:

1.3

6.350.2 Constructor & Destructor Documentation

- 6.350.2.1 **cms::InvalidDestinationException::InvalidDestinationException** () throw ()
- 6.350.2.2 **cms::InvalidDestinationException::InvalidDestinationException** (const **InvalidDestinationException** & *ex*) throw ()
- 6.350.2.3 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.350.2.4 **cms::InvalidDestinationException::InvalidDestinationException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.350.2.5 virtual **cms::InvalidDestinationException::~~InvalidDestinationException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidDestinationException.h**

6.351 decaf::security::InvalidKeyException Class Reference

#include <src/main/decaf/security/InvalidKeyException.h> Inheritance diagram for decaf::security::InvalidKeyException:

Public Member Functions

- **InvalidKeyException** () throw ()
Default Constructor.
- **InvalidKeyException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()
Copy Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidKeyException** (const std::exception *cause) throw ()
Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidKeyException** * clone () const
Clones this exception.
- virtual ~**InvalidKeyException** () throw ()

6.351.1 Constructor & Destructor Documentation

6.351.1.1 decaf::security::InvalidKeyException::InvalidKeyException () throw () [inline]

Default Constructor.

6.351.1.2 decaf::security::InvalidKeyException::InvalidKeyException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.351.1.3 decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.351.1.4 decaf::security::InvalidKeyException::InvalidKeyException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.351.1.5 decaf::security::InvalidKeyException::InvalidKeyException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.351.1.6 decaf::security::InvalidKeyException::InvalidKeyException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.351.1.7 `virtual decaf::security::InvalidKeyException::~~InvalidKeyException ()
throw () [inline, virtual]`

6.351.2 Member Function Documentation

6.351.2.1 `virtual InvalidKeyException* decaf::security::InvalidKeyException::clone
() const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from `decaf::security::KeyException` (p. 1957).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/InvalidKeyException.h`

6.352 decaf::nio::InvalidMarkException Class Reference

#include <src/main/decaf/nio/InvalidMarkException.h> Inheritance diagram for decaf::nio::InvalidMarkException:

Public Member Functions

- **InvalidMarkException** () throw ()
Default Constructor.
- **InvalidMarkException** (const lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidMarkException** (const InvalidMarkException &ex) throw ()
Copy Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidMarkException** (const std::exception *cause) throw ()
Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidMarkException** * clone () const
Clones this exception.
- virtual ~**InvalidMarkException** () throw ()

6.352.1 Constructor & Destructor Documentation

6.352.1.1 decaf::nio::InvalidMarkException::InvalidMarkException () throw () [inline]

Default Constructor.

6.352.1.2 decaf::nio::InvalidMarkException::InvalidMarkException (const lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex The Exception whose state data is to be copied into this Exception.

6.352.1.3 `decaf::nio::InvalidMarkException::InvalidMarkException (const InvalidMarkException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex The Exception whose state data is to be copied into this Exception.

6.352.1.4 `decaf::nio::InvalidMarkException::InvalidMarkException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.352.1.5 `decaf::nio::InvalidMarkException::InvalidMarkException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.352.1.6 `decaf::nio::InvalidMarkException::InvalidMarkException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.352.1.7 virtual decaf::nio::InvalidMarkException::~InvalidMarkException ()
throw () [inline, virtual]

6.352.2 Member Function Documentation

6.352.2.1 virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone ()
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new Exception instance that is a copy of this Exception.

Reimplemented from **decaf::lang::exceptions::IllegalStateException** (p. 1728).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**InvalidMarkException.h**

6.353 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

#include <src/main/cms/InvalidSelectorException.h> Inheritance diagram for cms::InvalidSelectorException:

Public Member Functions

- **InvalidSelectorException** () throw ()
- **InvalidSelectorException** (const **InvalidSelectorException** &ex) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidSelectorException** () throw ()

6.353.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Since:

1.3

6.353.2 Constructor & Destructor Documentation

6.353.2.1 cms::InvalidSelectorException::InvalidSelectorException () throw ()

6.353.2.2 cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & ex) throw ()

6.353.2.3 cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause) throw ()

6.353.2.4 cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()

6.353.2.5 virtual cms::InvalidSelectorException::~InvalidSelectorException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/InvalidSelectorException.h

6.354 decaf::lang::exceptions::InvalidStateException Class Reference

#include <src/main/decaf/lang/exceptions/InvalidStateException.h> Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

Public Member Functions

- **InvalidStateException** () throw ()
Default Constructor.
- **InvalidStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()
Copy Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidStateException** (const std::exception *cause) throw ()
Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidStateException** * clone () const
Clones this exception.
- virtual ~**InvalidStateException** () throw ()

6.354.1 Constructor & Destructor Documentation

6.354.1.1 decaf::lang::exceptions::InvalidStateException::InvalidStateException () throw () [inline]

Default Constructor.

6.354.1.2 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.354.1.3 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const InvalidStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.354.1.4 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.354.1.5 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.354.1.6 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.354.1.7 **virtual**
 decaf::lang::exceptions::InvalidStateException::~~InvalidStateException ()
 throw () [inline, virtual]

6.354.2 Member Function Documentation

6.354.2.1 **virtual InvalidStateException* de-**
 caf::lang::exceptions::InvalidStateException::clone () const
 [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InvalidStateException.h**

6.355 decaf::io::IOException Class Reference

#include <src/main/decaf/io/IOException.h> Inheritance diagram for decaf::io::IOException:

Public Member Functions

- **IOException** () throw ()
Default Constructor.
- **IOException** (const **lang::Exception** &ex) throw ()
Copy Constructor.
- **IOException** (const **IOException** &ex) throw ()
Copy Constructor.
- **IOException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IOException** (const std::exception *cause) throw ()
Constructor.
- **IOException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **IOException** * **clone** () const
Clones this exception.
- virtual ~**IOException** () throw ()

6.355.1 Constructor & Destructor Documentation

6.355.1.1 decaf::io::IOException::IOException () throw () [inline]

Default Constructor.

6.355.1.2 decaf::io::IOException::IOException (const **lang::Exception** & ex) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.355.1.3 decaf::io::IOException::IOException (const IOException & *ex*) throw ()
[inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.355.1.4 decaf::io::IOException::IOException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.355.1.5 decaf::io::IOException::IOException (const std::exception * *cause*) throw ()
[inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.355.1.6 decaf::io::IOException::IOException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.355.1.7 `virtual decaf::io::IOException::~~IOException () throw () [inline, virtual]`

6.355.2 Member Function Documentation

6.355.2.1 `virtual IOException* decaf::io::IOException::clone () const [inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an Exception that is a copy of this instance.

Reimplemented from `decaf::lang::Exception` (p. 1577).

Reimplemented in `decaf::io::EOFException` (p. 1573), `decaf::io::InterruptedIOException` (p. 1807), `decaf::io::UnsupportedEncodingException` (p. 3302), `decaf::io::UTFDataFormatException` (p. 3355), `decaf::net::BindException` (p. 723), `decaf::net::ConnectException` (p. 1130), `decaf::net::HttpRetryException` (p. 1719), `decaf::net::MalformedURLException` (p. 2093), `decaf::net::NoRouteToHostException` (p. 2419), `decaf::net::PortUnreachableException` (p. 2524), `decaf::net::ProtocolException` (p. 2669), `decaf::net::SocketException` (p. 2973), `decaf::net::SocketTimeoutException` (p. 2992), `decaf::net::UnknownHostException` (p. 3296), and `decaf::net::UnknownServiceException` (p. 3299).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`

6.356 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 3273) interface that performs marshaling of commands to IO streams.

#include <src/main/activemq/transport/IOTransport.h> Inheritance diagram for activemq::transport::IOTransport:

Public Member Functions

- **IOTransport** ()
Default Constructor.
- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
*Create an instance of this **Transport** (p. 3273) and assign its WireFormat instance at creation time.*
- virtual ~**IOTransport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported by this class - throws an exception.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **setInputStream** (decaf::io::DataInputStream *is)
Sets the input stream for in-coming commands.
- virtual void **setOutputStream** (decaf::io::DataOutputStream *os)
Sets the output stream for out-going commands.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.

- virtual void **stop** () throw (decaf::io::IOException)
*Stops the **Transport** (p. 3273), terminating any threads and stopping all read and write operations.*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **run** ()
Runs the polling thread.
- virtual **Transport * narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3273) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3273) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException)
reconnect to another location

6.356.1 Detailed Description

Implementation of the **Transport** (p. 3273) interface that performs marshaling of commands to IO streams. This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming commands. When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

6.356.2 Constructor & Destructor Documentation

6.356.2.1 activemq::transport::IOTransport::IOTransport ()

Default Constructor.

6.356.2.2 activemq::transport::IOTransport::IOTransport (const Pointer< wireformat::WireFormat > & wireFormat)

Create an instance of this **Transport** (p. 3273) and assign its WireFormat instance at creation time.

Parameters:

wireFormat Data encoder / decoder to use when reading and writing.

6.356.2.3 virtual `activemq::transport::IOTransport::~~IOTransport ()` [virtual]

6.356.3 Member Function Documentation

6.356.3.1 virtual `void activemq::transport::IOTransport::close () throw (decaf::io::IOException)` [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

IOException if errors occur.

Implements `decaf::io::Closeable` (p. 1019).

6.356.3.2 virtual `std::string activemq::transport::IOTransport::getRemoteAddress () const` [inline, virtual]

Returns:

the remote address for this connection

Implements `activemq::transport::Transport` (p. 3274).

6.356.3.3 virtual `TransportListener* activemq::transport::IOTransport::getTransportListener () const` [inline, virtual]

Gets the observer of asynchronous exceptions from this transport.

Returns:

The listener of transport events.

Implements `activemq::transport::Transport` (p. 3274).

6.356.3.4 virtual `bool activemq::transport::IOTransport::isClosed () const` [inline, virtual]

Has the `Transport` (p. 3273) been shutdown and no longer usable.

Returns:

true if the `Transport` (p. 3273)

Implements `activemq::transport::Transport` (p. 3275).

6.356.3.5 `virtual bool activemq::transport::IOTransport::isConnected () const`
[inline, virtual]

Is the **Transport** (p. 3273) Connected to its Broker.

Returns:

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3275).

6.356.3.6 `virtual bool activemq::transport::IOTransport::isFaultTolerant () const`
[inline, virtual]

Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 3273) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3275).

6.356.3.7 `virtual Transport* activemq::transport::IOTransport::narrow (const`
`std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3275).

6.356.3.8 `virtual void activemq::transport::IOTransport::oneway (const`
`Pointer< Command > & command) throw (decaf::io::IOException,`
`decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3276).

6.356.3.9 virtual void activemq::transport::IOTransport::reconnect
(const decaf::net::URI &uri *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

6.356.3.10 virtual Pointer<Response> activemq::transport::IOTransport::request
(const Pointer< Command > & *command*, unsigned
int *timeout*) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported by this class - throws an exception.

Parameters:

command the command to be sent.

timeout the time to wait for a response.

Returns:

the response to the command sent.

Exceptions:

UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 3276).

6.356.3.11 virtual Pointer<Response> activemq::transport::IOTransport::request
(const Pointer< Command > & *command*)
throw (decaf::io::IOException, de-
caaf::lang::exceptions::UnsupportedOperationException)
[virtual]

Not supported by this class - throws an exception.

Parameters:

command the command to be sent.

Returns:

the response to the command sent.

Exceptions:

UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 3277).

6.356.3.12 `virtual void activemq::transport::IOTransport::run ()` [virtual]

Runs the polling thread.

Implements `decaf::lang::Runnable` (p. 2816).

6.356.3.13 `virtual void activemq::transport::IOTransport::setInputStream (decaf::io::DataInputStream * is)` [inline, virtual]

Sets the input stream for in-coming commands.

Parameters:

is The input stream.

6.356.3.14 `virtual void activemq::transport::IOTransport::setOutputStream (decaf::io::DataOutputStream * os)` [inline, virtual]

Sets the output stream for out-going commands.

Parameters:

os The output stream.

6.356.3.15 `virtual void activemq::transport::IOTransport::setTransportListener (TransportListener * listener)` [inline, virtual]

Sets the observer of asynchronous exceptions from this transport.

Parameters:

listener the listener of transport events.

Implements `activemq::transport::Transport` (p. 3277).

6.356.3.16 `virtual void activemq::transport::IOTransport::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat)` [inline, virtual]

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode commands.

Implements `activemq::transport::Transport` (p. 3278).

6.356.3.17 `virtual void activemq::transport::IOTransport::start () throw (decaf::io::IOException)` [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands. If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

CMSException if an error occurs or if this transport has already been closed.

Implements **activemq::transport::Transport** (p. 3278).

6.356.3.18 `virtual void activemq::transport::IOTransport::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3273), terminating any threads and stopping all read and write operations.

Exceptions:

IOException if an error occurs while stopping the **Transport** (p. 3273).

Implements **activemq::transport::Transport** (p. 3278).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/IOTransport.h`

6.357 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 1830) type for generic collections API calls.

#include <src/main/decaf/lang/Iterable.h> Inheritance diagram for decaf::lang::Iterable< E >:

Public Member Functions

- virtual **~Iterable** ()
- virtual **decaf::util::Iterator< E > * iterator** ()=0
- virtual **decaf::util::Iterator< E > * iterator** () const =0

6.357.1 Detailed Description

template<typename E> class decaf::lang::Iterable< E >

Implementing this interface allows an object to be cast to an **Iterable** (p. 1830) type for generic collections API calls.

6.357.2 Constructor & Destructor Documentation

6.357.2.1 **template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable** ()
[inline, virtual]

6.357.3 Member Function Documentation

6.357.3.1 **template<typename E> virtual decaf::util::Iterator<E>***
decaf::lang::Iterable< E >::iterator () const [pure virtual]

Implemented in **decaf::util::PriorityQueue< E >** (p. 2575), **decaf::util::StlList< E >** (p. 3025), **decaf::util::StlSet< E >** (p. 3055), **decaf::util::StlList< CompositeTask * >** (p. 3025), **decaf::util::StlList< URI >** (p. 3025), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3025), **decaf::util::StlList< PrimitiveValueNode >** (p. 3025), **decaf::util::StlList< Pointer< Command > >** (p. 3025), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3025), **decaf::util::StlSet< transport::TransportListener * >** (p. 3055), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3055), and **decaf::util::StlSet< ActiveMQSession * >** (p. 3055).

6.357.3.2 **template<typename E> virtual decaf::util::Iterator<E>***
decaf::lang::Iterable< E >::iterator () [pure virtual]

Returns:

an iterator over a set of elements of type T.

Implemented in **decaf::util::PriorityQueue< E >** (p. 2575), **decaf::util::StlList< E >** (p. 3026), **decaf::util::StlSet< E >** (p. 3055), **decaf::util::StlList< CompositeTask * >**

(p. 3026), **decaf::util::StlList< URI >** (p. 3026), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3026), **decaf::util::StlList< PrimitiveValueNode >** (p. 3026), **decaf::util::StlList< Pointer< Command > >** (p. 3026), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3026), **decaf::util::StlSet< transport::TransportListener * >** (p. 3055), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3055), and **decaf::util::StlSet< ActiveMQSession * >** (p. 3055).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::clear()**, **decaf::util::AbstractCollection< Pointer< BackupTransport > >::contains()**, **decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy()**, **decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=()**, **decaf::util::AbstractCollection< Pointer< BackupTransport > >::remove()**, **decaf::util::AbstractSet< ActiveMQSession * >::removeAll()**, **decaf::util::AbstractCollection< Pointer< BackupTransport > >::removeAll()**, **decaf::util::AbstractCollection< Pointer< BackupTransport > >::retainAll()**, and **decaf::util::AbstractCollection< Pointer< BackupTransport > >::toArray()**.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Iterable.h`

6.358 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

#include <src/main/decaf/util/Iterator.h> Inheritance diagram for decaf::util::Iterator< T >:

Public Member Functions

- virtual **~Iterator** ()
- virtual T **next** ()=0 throw (lang::exceptions::NoSuchElementException)
Returns the next element in the iteration.
- virtual bool **hasNext** () const =0
Returns true if the iteration has more elements.
- virtual void **remove** ()=0 throw (lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException)
Removes from the underlying collection the last element returned by the iterator (optional operation).

6.358.1 Detailed Description

template<typename T> class decaf::util::Iterator< T >

Defines an object that can be used to iterate over the elements of a collection. The iterator provides a way to access and remove elements with well defined semantics.

6.358.2 Constructor & Destructor Documentation

6.358.2.1 template<typename T> virtual decaf::util::Iterator< T >::~~Iterator ()
[inline, virtual]

6.358.3 Member Function Documentation

6.358.3.1 template<typename T> virtual bool decaf::util::Iterator< T >::hasNext () const [pure virtual]

Returns true if the iteration has more elements. Returns false if the next call to next would result in an NoSuchElementException to be thrown.

6.358.3.2 template<typename T> virtual T decaf::util::Iterator< T >::next ()
throw (lang::exceptions::NoSuchElementException) [pure virtual]

Returns the next element in the iteration. Calling this method repeatedly until the **hasNext()** (p. 1832) method returns false will return each element in the underlying collection exactly once.

Returns:

next element in the iteration of elements

Exceptions:

NoSuchElementException - iteration has no more elements.

6.358.3.3 `template<typename T> virtual void decaf::util::Iterator< T
>::remove () throw (lang::exceptions::IllegalStateException,
lang::exceptions::UnsupportedOperationException) [pure virtual]`

Removes from the underlying collection the last element returned by the iterator (optional operation). This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

Exceptions:

UnsupportedOperationException - if the remove operation is not supported by this **Iterator** (p. 1832).

IllegalStateException - if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Iterator.h**

6.359 activemq::commands::JournalQueueAck Class Reference

#include <src/main/activemq/commands/JournalQueueAck.h> Inheritance diagram for activemq::commands::JournalQueueAck:

Public Member Functions

- **JournalQueueAck** ()
- virtual **~JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalQueueAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageAck** > & **getMessageAck** () const
- virtual **Pointer**< **MessageAck** > & **getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &messageAck)

Static Public Attributes

- static const unsigned char **ID_JOURNALQUEUEACK** = 52

Protected Member Functions

- **JournalQueueAck** (const **JournalQueueAck** &)
- **JournalQueueAck** & **operator=** (const **JournalQueueAck** &)

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageAck** > **messageAck**

6.359.1 Constructor & Destructor Documentation

- 6.359.1.1 **activemq::commands::JournalQueueAck::JournalQueueAck** (const JournalQueueAck &) [inline, protected]
- 6.359.1.2 **activemq::commands::JournalQueueAck::JournalQueueAck** ()
- 6.359.1.3 **virtual activemq::commands::JournalQueueAck::~~JournalQueueAck** () [virtual]

6.359.2 Member Function Documentation

- 6.359.2.1 **virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

- 6.359.2.2 **virtual void activemq::commands::JournalQueueAck::copyDataStructure** (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

- 6.359.2.3 **virtual bool activemq::commands::JournalQueueAck::equals** (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

- 6.359.2.4 **virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType** () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

- 6.359.2.5 **virtual** **Pointer**<**ActiveMQDestination**>& **activemq::commands::JournalQueueAck::getDestination** ()
[virtual]
- 6.359.2.6 **virtual const** **Pointer**<**ActiveMQDestination**>& **activemq::commands::JournalQueueAck::getDestination** () **const**
[virtual]
- 6.359.2.7 **virtual** **Pointer**<**MessageAck**>& **activemq::commands::JournalQueueAck::getMessageAck** ()
[virtual]
- 6.359.2.8 **virtual const** **Pointer**<**MessageAck**>& **activemq::commands::JournalQueueAck::getMessageAck** ()
const [virtual]
- 6.359.2.9 **JournalQueueAck**& **activemq::commands::JournalQueueAck::operator=**
(**const JournalQueueAck** &) [inline, protected]
- 6.359.2.10 **virtual void** **activemq::commands::JournalQueueAck::setDestination**
(**const Pointer**< **ActiveMQDestination** > & *destination*) [virtual]
- 6.359.2.11 **virtual void** **activemq::commands::JournalQueueAck::setMessageAck**
(**const Pointer**< **MessageAck** > & *messageAck*) [virtual]
- 6.359.2.12 **virtual std::string** **activemq::commands::JournalQueueAck::toString** ()
const [virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 718).

6.359.3 Field Documentation

- 6.359.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination`
[protected]
- 6.359.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID _JOURNALQUEUEACK = 52` [static]
- 6.359.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

6.360 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1838).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.360.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1838). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.360.2 Constructor & Destructor Documentation

6.360.2.1 `activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

6.360.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

6.360.3 Member Function Documentation

6.360.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.360.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.360.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.360.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.360.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.360.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.360.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h

6.361 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1842).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.361.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1842). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.361.2 Constructor & Destructor Documentation

6.361.2.1 `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

6.361.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

6.361.3 Member Function Documentation

6.361.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.361.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.361.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.361.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.361.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.361.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.361.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalQueueAckMarshaller.h**

6.362 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1846).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.362.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1846). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.362.2 Constructor & Destructor Documentation

6.362.2.1 `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

6.362.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

6.362.3 Member Function Documentation

6.362.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.362.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.362.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.362.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.362.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.362.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.362.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h

6.363 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1850).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.363.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1850). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.363.2 Constructor & Destructor Documentation

6.363.2.1 `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` [inline]

6.363.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` [inline, virtual]

6.363.3 Member Function Documentation

6.363.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.363.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.363.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.363.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.363.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.363.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.363.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalQueueAckMarshaller.h**

6.364 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1854).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.364.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1854). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.364.2 Constructor & Destructor Documentation

6.364.2.1 `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::JournalQueueAckMarshaller()` `[inline]`

6.364.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller()` `[inline, virtual]`

6.364.3 Member Function Documentation

6.364.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::createObject() const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.364.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.364.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.364.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.364.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.364.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.364.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h

6.365 activemq::commands::JournalTopicAck Class Reference

#include <src/main/activemq/commands/JournalTopicAck.h> Inheritance diagram for activemq::commands::JournalTopicAck:

Public Member Functions

- **JournalTopicAck** ()
- virtual **~JournalTopicAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTopicAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src) const
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual long long **getMessageSequenceId** () const
- virtual void **setMessageSequenceId** (long long messageSequenceId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

Static Public Attributes

- static const unsigned char **ID_JOURNALTOPICACK** = 50

Protected Member Functions

- **JournalTopicAck** (const **JournalTopicAck** &)
- **JournalTopicAck** & operator= (const **JournalTopicAck** &)

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageId** > **messageId**
- long long **messageSequenceId**
- std::string **subscriptionName**
- std::string **clientId**
- **Pointer**< **TransactionId** > **transactionId**

6.365.1 Constructor & Destructor Documentation

- 6.365.1.1** **activemq::commands::JournalTopicAck::JournalTopicAck** (const **JournalTopicAck** &) [inline, protected]
- 6.365.1.2** **activemq::commands::JournalTopicAck::JournalTopicAck** ()
- 6.365.1.3** **virtual activemq::commands::JournalTopicAck::~~JournalTopicAck** () [virtual]

6.365.2 Member Function Documentation

- 6.365.2.1** **virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

- 6.365.2.2** **virtual void activemq::commands::JournalTopicAck::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.365.2.3 `virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.365.2.4 `virtual std::string& activemq::commands::JournalTopicAck::getClientId ()` [virtual]

6.365.2.5 `virtual const std::string& activemq::commands::JournalTopicAck::getClientId () const` [virtual]

6.365.2.6 `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

-
- 6.365.2.7 virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () [virtual]
- 6.365.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () const [virtual]
- 6.365.2.9 virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () [virtual]
- 6.365.2.10 virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () const [virtual]
- 6.365.2.11 virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId () const [virtual]
- 6.365.2.12 virtual std::string& activemq::commands::JournalTopicAck::getSubscriptionName () [virtual]
- 6.365.2.13 virtual const std::string& activemq::commands::JournalTopicAck::getSubscriptionName () const [virtual]
- 6.365.2.14 virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () [virtual]
- 6.365.2.15 virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () const [virtual]
- 6.365.2.16 JournalTopicAck& activemq::commands::JournalTopicAck::operator= (const JournalTopicAck &) [inline, protected]
- 6.365.2.17 virtual void activemq::commands::JournalTopicAck::setClientId (const std::string & *clientId*) [virtual]
- 6.365.2.18 virtual void activemq::commands::JournalTopicAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.365.2.19 virtual void activemq::commands::JournalTopicAck::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.365.2.20 virtual void activemq::commands::JournalTopicAck::setMessageSequenceId (long long *messageSequenceId*) [virtual]
- 6.365.2.21 virtual void activemq::commands::JournalTopicAck::setSubscriptionName (const std::string & *subscriptionName*) [virtual]
-
- 6.365.2.22 virtual void activemq::commands::JournalTopicAck::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]
- 6.365.2.23 virtual std::string activemq::commands::JournalTopicAck::toString () const [virtual]
-

Returns a string containing the information for this **DataSet** (p.1461) such as its type

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 718).

6.365.3 Field Documentation

- 6.365.3.1** `std::string activemq::commands::JournalTopicAck::clientId` [protected]
- 6.365.3.2** `Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination` [protected]
- 6.365.3.3** `const unsigned char activemq::commands::JournalTopicAck::ID_ - JOURNALTOPICACK = 50` [static]
- 6.365.3.4** `Pointer<MessageId> activemq::commands::JournalTopicAck::messageId` [protected]
- 6.365.3.5** `long long activemq::commands::JournalTopicAck::messageSequenceId` [protected]
- 6.365.3.6** `std::string activemq::commands::JournalTopicAck::subscriptionName` [protected]
- 6.365.3.7** `Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTopicAck.h`

6.366 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1863).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.366.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1863). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.366.2 Constructor & Destructor Documentation

6.366.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.366.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.366.3 Member Function Documentation

6.366.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.366.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.366.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.366.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.366.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.366.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.366.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTopicAckMarshaller.h**

6.367 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1867).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.367.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1867). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.367.2 Constructor & Destructor Documentation

6.367.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.367.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.367.3 Member Function Documentation

6.367.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.367.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.367.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.367.3.4 virtual void **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.367.3.5 virtual int **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.367.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.367.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTopicAckMarshaller.h**

6.368 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1871).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.368.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1871). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.368.2 Constructor & Destructor Documentation

6.368.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.368.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.368.3 Member Function Documentation

6.368.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.368.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.368.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.368.3.4 virtual void **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.368.3.5 virtual int **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.368.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.368.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTopicAckMarshaller.h**

6.369 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1875).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.369.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1875). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.369.2 Constructor & Destructor Documentation

6.369.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.369.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.369.3 Member Function Documentation

6.369.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.369.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.369.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.369.3.4 virtual void **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.369.3.5 virtual int **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.369.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.369.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTopicAckMarshaller.h**

6.370 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1879).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h>Inheritance
 diagram for activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn,
utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (de-
 caf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut,
utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
 throw (decaf::io::IOException)
Write a object instance to data output stream.

6.370.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1879). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.370.2 Constructor & Destructor Documentation

6.370.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.370.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.370.3 Member Function Documentation

6.370.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.370.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.370.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.370.3.4 virtual void **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.370.3.5 virtual int **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.370.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.370.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTopicAckMarshaller.h**

6.371 activemq::commands::JournalTrace Class Reference

#include <src/main/activemq/commands/JournalTrace.h> Inheritance diagram for activemq::commands::JournalTrace:

Public Member Functions

- **JournalTrace** ()
- virtual **~JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTrace * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRACE** = 53

Protected Member Functions

- **JournalTrace** (const **JournalTrace** &)
- **JournalTrace** & **operator=** (const **JournalTrace** &)

Protected Attributes

- std::string **message**

6.371.1 Constructor & Destructor Documentation

6.371.1.1 `activemq::commands::JournalTrace::JournalTrace (const JournalTrace &) [inline, protected]`

6.371.1.2 `activemq::commands::JournalTrace::JournalTrace ()`

6.371.1.3 `virtual activemq::commands::JournalTrace::~~JournalTrace () [virtual]`

6.371.2 Member Function Documentation

6.371.2.1 `virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.371.2.2 `virtual void activemq::commands::JournalTrace::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

6.371.2.3 `virtual bool activemq::commands::JournalTrace::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

6.371.2.4 `virtual unsigned char activemq::commands::JournalTrace::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

- 6.371.2.5** `virtual std::string& activemq::commands::JournalTrace::getMessage ()`
[virtual]
- 6.371.2.6** `virtual const std::string& activemq::commands::JournalTrace::getMessage () const` [virtual]
- 6.371.2.7** `JournalTrace& activemq::commands::JournalTrace::operator= (const JournalTrace &)` [inline, protected]
- 6.371.2.8** `virtual void activemq::commands::JournalTrace::setMessage (const std::string & message)` [virtual]
- 6.371.2.9** `virtual std::string activemq::commands::JournalTrace::toString () const`
[virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 718).

6.371.3 Field Documentation

- 6.371.3.1** `const unsigned char activemq::commands::JournalTrace::ID_ - JOURNALTRACE = 53` [static]
- 6.371.3.2** `std::string activemq::commands::JournalTrace::message` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

6.372 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1886).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.372.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1886). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.372.2 Constructor & Destructor Documentation

6.372.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

6.372.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

6.372.3 Member Function Documentation

6.372.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.372.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.372.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.372.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.372.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.372.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.372.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**

6.373 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1890).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.373.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1890). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.373.2 Constructor & Destructor Documentation

6.373.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

6.373.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

6.373.3 Member Function Documentation

6.373.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.373.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.373.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.373.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.373.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.373.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.373.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTraceMarshaller.h**

6.374 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1894).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.374.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1894). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.374.2 Constructor & Destructor Documentation

6.374.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarshaller()` `[inline]`

6.374.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~~JournalTraceMarshaller()` `[inline, virtual]`

6.374.3 Member Function Documentation

6.374.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.374.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.374.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.374.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.374.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.374.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.374.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTraceMarshaller.h**

6.375 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1898).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.375.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1898). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.375.2 Constructor & Destructor Documentation

6.375.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

6.375.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

6.375.3 Member Function Documentation

6.375.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.375.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.375.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.375.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.375.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.375.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.375.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**

6.376 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1902).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.376.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1902). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.376.2 Constructor & Destructor Documentation

6.376.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::JournalTraceMarshaller()` [inline]

6.376.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::~~JournalTraceMarshaller()` [inline, virtual]

6.376.3 Member Function Documentation

6.376.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.376.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.376.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.376.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.376.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.376.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.376.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTraceMarshaller.h**

6.377 activemq::commands::JournalTransaction Class Reference

#include <src/main/activemq/commands/JournalTransaction.h> Inheritance diagram for activemq::commands::JournalTransaction:

Public Member Functions

- **JournalTransaction** ()
- virtual **~JournalTransaction** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTransaction * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool wasPrepared)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRANSACTION** = 54

Protected Member Functions

- **JournalTransaction** (const **JournalTransaction** &)
- **JournalTransaction** & **operator=** (const **JournalTransaction** &)

Protected Attributes

- `Pointer< TransactionId > transactionId`
- unsigned char `type`
- bool `wasPrepared`

6.377.1 Constructor & Destructor Documentation

6.377.1.1 `activemq::commands::JournalTransaction::JournalTransaction (const JournalTransaction &) [inline, protected]`

6.377.1.2 `activemq::commands::JournalTransaction::JournalTransaction ()`

6.377.1.3 `virtual activemq::commands::JournalTransaction::~~JournalTransaction () [virtual]`

6.377.2 Member Function Documentation

6.377.2.1 `virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.377.2.2 `virtual void activemq::commands::JournalTransaction::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

6.377.2.3 `virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

6.377.2.4 virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.377.2.5 virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () [virtual]

6.377.2.6 virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () const [virtual]

6.377.2.7 virtual unsigned char activemq::commands::JournalTransaction::getType () const [virtual]

6.377.2.8 virtual bool activemq::commands::JournalTransaction::getWasPrepared () const [virtual]

6.377.2.9 JournalTransaction& activemq::commands::JournalTransaction::operator= (const JournalTransaction &) [inline, protected]

6.377.2.10 virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.377.2.11 virtual void activemq::commands::JournalTransaction::setType (unsigned char *type*) [virtual]

6.377.2.12 virtual void activemq::commands::JournalTransaction::setWasPrepared (bool *wasPrepared*) [virtual]

6.377.2.13 virtual std::string activemq::commands::JournalTransaction::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 718).

6.377.3 Field Documentation

- 6.377.3.1 `const unsigned char activemq::commands::JournalTransaction::ID_ - JOURNALTRANSACTION = 54` [static]
- 6.377.3.2 `Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId` [protected]
- 6.377.3.3 `unsigned char activemq::commands::JournalTransaction::type` [protected]
- 6.377.3.4 `bool activemq::commands::JournalTransaction::wasPrepared` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTransaction.h`

6.378 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1910).

#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.378.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1910). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.378.2 Constructor & Destructor Documentation

6.378.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTrans
() [inline]`

6.378.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~~JournalTra
() [inline, virtual]`

6.378.3 Member Function Documentation

6.378.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.378.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.378.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.378.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.378.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.378.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.378.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**

6.379 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1914).

#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.379.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1914). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.379.2 Constructor & Destructor Documentation

6.379.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTrans
() [inline]`

6.379.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~~JournalTra
() [inline, virtual]`

6.379.3 Member Function Documentation

6.379.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.379.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.379.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.379.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.379.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.379.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.379.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTransactionMarshaller.h**

6.380 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1918).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.380.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1918). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.380.2 Constructor & Destructor Documentation

6.380.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::JournalTrans
() [inline]`

6.380.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::~~JournalTra
() [inline, virtual]`

6.380.3 Member Function Documentation

6.380.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.380.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.380.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.380.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.380.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.380.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.380.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTransactionMarshaller.h**

6.381 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1922).

#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.381.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1922). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.381.2 Constructor & Destructor Documentation

6.381.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTrans
() [inline]`

6.381.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~~JournalTra
() [inline, virtual]`

6.381.3 Member Function Documentation

6.381.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.381.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.381.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.381.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.381.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.381.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.381.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**

6.382 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1926).

#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.382.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1926). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.382.2 Constructor & Destructor Documentation

6.382.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::JournalTrans
() [inline]`

6.382.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::~~JournalTra
() [inline, virtual]`

6.382.3 Member Function Documentation

6.382.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.382.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::getDataStructu
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.382.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.382.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.382.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.382.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.382.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTransactionMarshaller.h**

6.383 activemq::commands::KeepAliveInfo Class Reference

#include <src/main/activemq/commands/KeepAliveInfo.h> Inheritance diagram for activemq::commands::KeepAliveInfo:

Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **KeepAliveInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _KEEPALIVEINFO** = 10

Protected Member Functions

- **KeepAliveInfo** (const **KeepAliveInfo** &)
- **KeepAliveInfo** & **operator=** (const **KeepAliveInfo** &)

6.383.1 Constructor & Destructor Documentation

6.383.1.1 `activemq::commands::KeepAliveInfo::KeepAliveInfo (const KeepAliveInfo &) [inline, protected]`

6.383.1.2 `activemq::commands::KeepAliveInfo::KeepAliveInfo ()`

6.383.1.3 `virtual activemq::commands::KeepAliveInfo::~~KeepAliveInfo () [virtual]`

6.383.2 Member Function Documentation

6.383.2.1 `virtual KeepAliveInfo* activemq::commands::KeepAliveInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.383.2.2 `virtual void activemq::commands::KeepAliveInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.383.2.3 `virtual bool activemq::commands::KeepAliveInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.383.2.4 `virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

6.383.2.5 **virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo ()**
const [inline, virtual]

Returns:

an answer of true to the **isKeepAliveInfo()** (p. 1932) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 653).

6.383.2.6 **KeepAliveInfo& activemq::commands::KeepAliveInfo::operator= (const**
KeepAliveInfo &) [inline, protected]

6.383.2.7 **virtual std::string activemq::commands::KeepAliveInfo::toString () const**
[virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.383.2.8 **virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit**
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.383.3 Field Documentation

6.383.3.1 **const unsigned char activemq::commands::KeepAliveInfo::ID _-**
KEEPALIVEINFO = 10 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**KeepAliveInfo.h**

6.384 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1933).

#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.384.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1933). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.384.2 Constructor & Destructor Documentation

6.384.2.1 `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.384.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.384.3 Member Function Documentation

6.384.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.384.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.384.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.384.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.384.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.384.3.6 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.384.3.7 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h

6.385 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1937).

#include <src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.385.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1937). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.385.2 Constructor & Destructor Documentation

6.385.2.1 `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.385.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.385.3 Member Function Documentation

6.385.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.385.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.385.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.385.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.385.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.385.3.6 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.385.3.7 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h

6.386 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1941).

#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.386.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1941). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.386.2 Constructor & Destructor Documentation

6.386.2.1 `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.386.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.386.3 Member Function Documentation

6.386.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.386.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.386.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.386.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.386.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.386.3.6 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 661).

6.386.3.7 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 662).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h`

6.387 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1945).

#include <src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.387.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1945). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.387.2 Constructor & Destructor Documentation

6.387.2.1 `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.387.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.387.3 Member Function Documentation

6.387.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.387.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.387.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 672).

6.387.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 673).

6.387.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 674).

6.387.3.6 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.387.3.7 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h

6.388 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1949).

#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.388.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1949). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.388.2 Constructor & Destructor Documentation

6.388.2.1 `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.388.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.388.3 Member Function Documentation

6.388.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.388.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.388.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.388.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.388.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.388.3.6 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.388.3.7 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

6.389 decaf::security::Key Class Reference

The **Key** (p. 1953) interface is the top-level interface for all keys.

#include <src/main/decaf/security/Key.h> Inheritance diagram for decaf::security::Key:

Public Member Functions

- virtual **~Key** ()
- virtual std::string **getAlgorithm** () const =0
Returns the standard algorithm name for this key.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
Provides the key in its primary encoding format, or nothing if this key does not support encoding.
- virtual std::string **getFormat** () const =0
Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

6.389.1 Detailed Description

The **Key** (p. 1953) interface is the top-level interface for all keys. It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the `getAlgorithm` method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 SubjectPublicKeyInfo or PKCS#8), and is returned using the `getEncoded` method. Note: The syntax of the ASN.1 type SubjectPublicKeyInfo is defined as follows:

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm AlgorithmIdentifier,
    subjectPublicKey BIT STRING }
AlgorithmIdentifier ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm OPTIONAL }
```

For more information, see RFC 2459: Internet X.509 Public **Key** (p. 1953) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

6.389.2 Constructor & Destructor Documentation

6.389.2.1 `virtual decaf::security::Key::~~Key () [inline, virtual]`

6.389.3 Member Function Documentation

6.389.3.1 `virtual std::string decaf::security::Key::getAlgorithm () const [pure virtual]`

Returns the standard algorithm name for this key. For example, "DSA" would indicate that this key is a DSA key.

Returns:

the name of the algorithm associated with this key.

6.389.3.2 `virtual void decaf::security::Key::getEncoded (std::vector< unsigned char > & output) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

Parameters:

output Receives the encoded key, or nothing if the key does not support encoding.

6.389.3.3 `virtual std::string decaf::security::Key::getFormat () const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding. The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is `SubjectPublicKeyInfo`, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is `PrivateKeyInfo`, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

Returns:

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Key.h`

6.390 decaf::security::KeyException Class Reference

#include <src/main/decaf/security/KeyException.h> Inheritance diagram for decaf::security::KeyException:

Public Member Functions

- **KeyException** () throw ()
Default Constructor.
- **KeyException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **KeyException** (const **KeyException** &ex) throw ()
Copy Constructor.
- **KeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **KeyException** (const std::exception *cause) throw ()
Constructor.
- **KeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **KeyException** * clone () const
Clones this exception.
- virtual ~**KeyException** () throw ()

6.390.1 Constructor & Destructor Documentation

6.390.1.1 decaf::security::KeyException::KeyException () throw () [inline]

Default Constructor.

6.390.1.2 decaf::security::KeyException::KeyException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.390.1.3 decaf::security::KeyException::KeyException (const KeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.390.1.4 decaf::security::KeyException::KeyException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.390.1.5 decaf::security::KeyException::KeyException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.390.1.6 decaf::security::KeyException::KeyException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.390.1.7 virtual decaf::security::KeyException::~~KeyException () throw ()
[inline, virtual]

6.390.2 Member Function Documentation

6.390.2.1 virtual KeyException* decaf::security::KeyException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1708).

Reimplemented in **decaf::security::InvalidKeyException** (p. 1812).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**KeyException.h**

6.391 activemq::commands::LastPartialCommand Class Reference

#include <src/main/activemq/commands/LastPartialCommand.h> Inheritance diagram for activemq::commands::LastPartialCommand:

Public Member Functions

- **LastPartialCommand** ()
- virtual **~LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **LastPartialCommand * cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*

Static Public Attributes

- static const unsigned char **ID_LASTPARTIALCOMMAND** = 61

Protected Member Functions

- **LastPartialCommand** (const **LastPartialCommand** &)
- **LastPartialCommand** & **operator=** (const **LastPartialCommand** &)

6.391.1 Constructor & Destructor Documentation

6.391.1.1 `activemq::commands::LastPartialCommand::LastPartialCommand (const LastPartialCommand &) [inline, protected]`

6.391.1.2 `activemq::commands::LastPartialCommand::LastPartialCommand ()`

6.391.1.3 `virtual
activemq::commands::LastPartialCommand::~~LastPartialCommand ()
[virtual]`

6.391.2 Member Function Documentation

6.391.2.1 `virtual LastPartialCommand* ac-
tivemq::commands::LastPartialCommand::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::PartialCommand` (p. 2474).

6.391.2.2 `virtual void ac-
tivemq::commands::LastPartialCommand::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::PartialCommand` (p. 2474).

6.391.2.3 `virtual bool activemq::commands::LastPartialCommand::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::PartialCommand` (p. 2474).

6.391.2.4 `virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Reimplemented from **activemq::commands::PartialCommand** (p. 2475).

6.391.2.5 `LastPartialCommand& activemq::commands::LastPartialCommand::operator= (const LastPartialCommand &) [inline, protected]`

6.391.2.6 `virtual std::string activemq::commands::LastPartialCommand::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 2475).

6.391.3 Field Documentation

6.391.3.1 `const unsigned char activemq::commands::LastPartialCommand::ID_ - LASTPARTIALCOMMAND = 61 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LastPartialCommand.h`

6.392 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1961).

#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.392.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1961).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.392.2 Constructor & Destructor Documentation

6.392.2.1 `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.392.2.2 `virtual activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.392.3 Member Function Documentation

6.392.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2478).

6.392.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2478).

6.392.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2478).

6.392.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2479).

6.392.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2479).

6.392.3.6 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2480).

6.392.3.7 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2480).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**

6.393 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1965).

#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.393.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1965).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.393.2 Constructor & Destructor Documentation

6.393.2.1 `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.393.2.2 `virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.393.3 Member Function Documentation

6.393.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2490).

6.393.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2490).

6.393.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2490).

6.393.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2491).

6.393.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2491).

6.393.3.6 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2492).

6.393.3.7 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2492).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**

6.394 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1969).

#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.394.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1969).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.394.2 Constructor & Destructor Documentation

6.394.2.1 `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.394.2.2 `virtual activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.394.3 Member Function Documentation

6.394.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2482).

6.394.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2482).

6.394.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2482).

6.394.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2483).

6.394.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2483).

6.394.3.6 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2484).

6.394.3.7 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2484).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LastPartialCommandMarshaller.h**

6.395 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1973).

#include <src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.395.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1973).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.395.2 Constructor & Destructor Documentation

6.395.2.1 `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.395.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.395.3 Member Function Documentation

6.395.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2486).

6.395.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2486).

6.395.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2486).

6.395.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2487).

6.395.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2487).

6.395.3.6 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2488).

6.395.3.7 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2488).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LastPartialCommandMarshaller.h**

6.396 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1977).

#include <src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.396.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1977).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.396.2 Constructor & Destructor Documentation

6.396.2.1 `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.396.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` [inline, virtual]

6.396.3 Member Function Documentation

6.396.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2494).

6.396.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2494).

6.396.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2494).

6.396.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2495).

6.396.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2495).

6.396.3.6 virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2496).

6.396.3.7 virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2496).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h

6.397 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 1981) **Comparator** (p. 1086) that compares to elements to determine if the first is less than the second.

#include <src/main/decaf/util/comparators/Less.h> Inheritance diagram for decaf::util::comparators::Less< E >:

Public Member Functions

- **Less** ()
- virtual **~Less** ()
- virtual bool **operator**() (const E &left, const E &right) const
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1086) to be passed to an STL **Map** (p. 2094) for use as the sorting criteria.*
- virtual int **compare** (const E &o1, const E &o2) const
Compares its two arguments for order.

6.397.1 Detailed Description

template<typename E> class decaf::util::comparators::Less< E >

Simple **Less** (p. 1981) **Comparator** (p. 1086) that compares to elements to determine if the first is less than the second. This can be used in **Collection** (p. 1054) classes to sort elements according to their natural ordering. By design the Comparator's compare function return more information about comparison than the STL binary function's boolean compare operator. In this case the compare method will return

Since:

1.0

6.397.2 Constructor & Destructor Documentation

6.397.2.1 template<typename E > decaf::util::comparators::Less< E >::Less ()
[inline]

6.397.2.2 template<typename E > virtual decaf::util::comparators::Less< E >::~~Less () [inline, virtual]

6.397.3 Member Function Documentation

6.397.3.1 template<typename E > virtual int decaf::util::comparators::Less< E >::compare (const E & o1, const E & o2) const [inline, virtual]

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that $\text{sgn}(\text{compare}(x, y)) == -\text{sgn}(\text{compare}(y, x))$ for all x and y . (This implies that $\text{compare}(x, y)$ must throw an exception if and only if $\text{compare}(y, x)$ throws an exception.)

The implementor must also ensure that the relation is transitive: $((\text{compare}(x, y) > 0) \ \&\& \ (\text{compare}(y, z) > 0))$ implies $\text{compare}(x, z) > 0$.

Finally, the implementer must ensure that $\text{compare}(x, y) == 0$ implies that $\text{sgn}(\text{compare}(x, z)) == \text{sgn}(\text{compare}(y, z))$ for all z .

It is generally the case, but not strictly required that $(\text{compare}(x, y) == 0) == (x == y)$. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters:

- o1* - the first object to be compared
- o2* - the second object to be compared

Returns:

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements **decaf::util::Comparator**< **E** > (p.1086).

6.397.3.2 `template<typename E> virtual bool decaf::util::comparators::Less< E >::operator() (const E & left, const E & right) const` [inline, virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p.1086) to be passed to an STL **Map** (p.2094) for use as the sorting criteria.

Parameters:

- left* - the Left hand side operand.
- right* - the Right hand side operand.

Returns:

true if the vale of left is less than the value of right.

Implements **decaf::util::Comparator**< **E** > (p.1087).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/comparators/`**Less.h**

6.398 std::less< decaf::lang::Pointer< T > > Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/Pointer.h>Inheritance    diagram    for    std::less<
decaf::lang::Pointer< T > >:
```

Public Member Functions

- bool **operator()** (const decaf::lang::Pointer< T > &left, const decaf::lang::Pointer< T > &right) const

6.398.1 Detailed Description

```
template<typename T> struct std::less< decaf::lang::Pointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.398.2 Member Function Documentation

6.398.2.1 template<typename T > bool std::less< decaf::lang::Pointer< T > >::operator() (const decaf::lang::Pointer< T > & *left*, const decaf::lang::Pointer< T > & *right*) const [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.399 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

#include <src/main/decaf/util/List.h> Inheritance diagram for decaf::util::List< E >:

Public Member Functions

- virtual **~List** ()
- virtual **ListIterator**< E > * **listIterator** ()=0
- virtual **ListIterator**< E > * **listIterator** () const =0
- virtual **ListIterator**< E > * **listIterator** (std::size_t index)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **indexOf** (const E &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)
Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual std::size_t **lastIndexOf** (const E &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)
Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
- virtual E **get** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Gets the element contained at position passed.
- virtual E **set** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Replaces the element at the specified position in this list with the specified element.
- virtual void **add** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)
Inserts the specified element at the specified position in this list.
- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)
Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
- virtual E **remove** (std::size_t index)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)
Removes the element at the specified position in this list.

6.399.1 Detailed Description

template<typename E> class decaf::util::List< E >

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements *e1* and *e2* such that *e1.equals(e2)*, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

6.399.2 Constructor & Destructor Documentation

6.399.2.1 **template<typename E> virtual decaf::util::List< E >::~~List ()** [inline, virtual]

6.399.3 Member Function Documentation

6.399.3.1 **template<typename E> virtual void decaf::util::List< E >::add (std::size_t *index*, const E & *element*)** throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters:

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 3022), **decaf::util::StlList< CompositeTask * >** (p. 3022), **decaf::util::StlList< URI >** (p. 3022), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3022), **decaf::util::StlList< PrimitiveValueNode >** (p. 3022), **decaf::util::StlList< Pointer< Command > >** (p. 3022), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3022).

6.399.3.2 **template<typename E> virtual bool decaf::util::List< E >::addAll (std::size_t *index*, const Collection< E > & *source*)** throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Inserts all of the elements in the specified collection into this list at the specified position (optional operation). Shifts the element currently at that position (if any) and any subsequent elements to

the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters:

index The index at which to insert the first element from the specified collection
source The **Collection** (p. 1054) containing elements to be added to this list

Returns:

true if this list changed as a result of the call

Exceptions:

IndexOutOfBoundsException - if the index is greater than size
UnsupportedOperationException - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 3023), **decaf::util::StlList< CompositeTask * >** (p. 3023), **decaf::util::StlList< URI >** (p. 3023), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3023), **decaf::util::StlList< PrimitiveValueNode >** (p. 3023), **decaf::util::StlList< Pointer< Command > >** (p. 3023), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3023).

6.399.3.3 `template<typename E> virtual E decaf::util::List< E >::get (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Gets the element contained at position passed.

Parameters:

index - position to get

Returns:

value at index

Implemented in **decaf::util::StlList< E >** (p. 3024), **decaf::util::StlList< CompositeTask * >** (p. 3024), **decaf::util::StlList< URI >** (p. 3024), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3024), **decaf::util::StlList< PrimitiveValueNode >** (p. 3024), **decaf::util::StlList< Pointer< Command > >** (p. 3024), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3024).

6.399.3.4 `template<typename E> virtual std::size_t decaf::util::List< E >::indexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the lowest index i such that get(i) == value, or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the first occurrence of the specified element in this list,

Exceptions:

NoSuchElementException if value is not in the list

Implemented in `decaf::util::StlList< E >` (p. 3025), `decaf::util::StlList< CompositeTask * >` (p. 3025), `decaf::util::StlList< URI >` (p. 3025), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3025), `decaf::util::StlList< PrimitiveValueNode >` (p. 3025), `decaf::util::StlList< Pointer< Command > >` (p. 3025), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3025).

6.399.3.5 `template<typename E> virtual size_t decaf::util::List< E >::lastIndexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the last occurrence of the specified element in this list.

Exceptions:

NoSuchElementException if value is not in the list

Implemented in `decaf::util::StlList< E >` (p. 3026), `decaf::util::StlList< CompositeTask * >` (p. 3026), `decaf::util::StlList< URI >` (p. 3026), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3026), `decaf::util::StlList< PrimitiveValueNode >` (p. 3026), `decaf::util::StlList< Pointer< Command > >` (p. 3026), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3026).

6.399.3.6 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Implemented in `decaf::util::StlList< E >` (p. 3026), `decaf::util::StlList< CompositeTask * >` (p. 3026), `decaf::util::StlList< URI >` (p. 3026), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3026), `decaf::util::StlList< PrimitiveValueNode >` (p. 3026), `decaf::util::StlList< Pointer< Command > >` (p. 3026), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3026).

6.399.3.7 `template<typename E> virtual ListIterator<E>*
decaf::util::List< E >::listIterator (std::size_t index) throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Parameters:

index index of first element to be returned from the list iterator (by a call to the next method).

Returns:

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions:

IndexOutOfBoundsException if the index is out of range (`index < 0 || index > size()` (p. 1062))

Implemented in `decaf::util::StlList< E >` (p. 3026), `decaf::util::StlList< CompositeTask * >` (p. 3026), `decaf::util::StlList< URI >` (p. 3026), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3026), `decaf::util::StlList< PrimitiveValueNode >` (p. 3026), `decaf::util::StlList< Pointer< Command > >` (p. 3026), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3026).

6.399.3.8 `template<typename E> virtual ListIterator<E>* decaf::util::List< E
>::listIterator () const [pure virtual]`

Implemented in `decaf::util::StlList< E >` (p. 3027), `decaf::util::StlList< CompositeTask * >` (p. 3027), `decaf::util::StlList< URI >` (p. 3027), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3027), `decaf::util::StlList< PrimitiveValueNode >` (p. 3027), `decaf::util::StlList< Pointer< Command > >` (p. 3027), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3027).

6.399.3.9 `template<typename E> virtual ListIterator<E>* decaf::util::List< E
>::listIterator () [pure virtual]`

Returns:

a list iterator over the elements in this list (in proper sequence).

Implemented in `decaf::util::StlList< E >` (p. 3027), `decaf::util::StlList< CompositeTask * >` (p. 3027), `decaf::util::StlList< URI >` (p. 3027), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3027), `decaf::util::StlList< PrimitiveValueNode >` (p. 3027), `decaf::util::StlList< Pointer< Command > >` (p. 3027), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3027).

6.399.3.10 `template<typename E> virtual E decaf::util::List< E >::remove (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters:

index - the index of the element to be removed

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3027), `decaf::util::StlList< CompositeTask * >` (p. 3027), `decaf::util::StlList< URI >` (p. 3027), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3027), `decaf::util::StlList< PrimitiveValueNode >` (p. 3027), `decaf::util::StlList< Pointer< Command > >` (p. 3027), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3027).

6.399.3.11 `template<typename E> virtual E decaf::util::List< E >::set (std::size_t index, const E & element) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters:

index - index of the element to replace

element - element to be stored at the specified position

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

Implemented in `decaf::util::StlList< E >` (p. 3028), `decaf::util::StlList< CompositeTask * >` (p. 3028), `decaf::util::StlList< URI >` (p. 3028), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3028), `decaf::util::StlList< PrimitiveValueNode >` (p. 3028), `decaf::util::StlList< Pointer< Command > >` (p. 3028), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3028).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/List.h`

6.400 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h> Inheritance diagram for
decaf::util::ListIterator< E >:
```

Public Member Functions

- virtual **~ListIterator** ()
- virtual void **add** (const E &e)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into the list (optional operation).
- virtual void **set** (const E &e)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Replaces the last element returned by next or previous with the specified element (optional operation).
- virtual bool **hasPrevious** () const =0
Returns true if this list iterator has more elements when traversing the list in the reverse direction.
- virtual E **previous** ()=0
Returns the previous element in the list.
- virtual int **nextIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to next.
- virtual int **previousIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to previous.

6.400.1 Detailed Description

```
template<typename E> class decaf::util::ListIterator< E >
```

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. Note that the **remove()** (p. 1833) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 1832) or **previous()** (p. 1992).

6.400.2 Constructor & Destructor Documentation

6.400.2.1 `template<typename E > virtual decaf::util::ListIterator< E >::~~ListIterator () [inline, virtual]`

6.400.3 Member Function Documentation

6.400.3.1 `template<typename E > virtual void decaf::util::ListIterator< E >::add (const E & e) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]`

Inserts the specified element into the list (optional operation). The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

Parameters:

e - the element to insert.

Exceptions:

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

6.400.3.2 `template<typename E > virtual bool decaf::util::ListIterator< E >::hasPrevious () const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction. (In other words, returns true if previous would return an element rather than throwing an exception.)

Returns:

true if the list iterator has more elements when traversing the list in the reverse direction.

6.400.3.3 `template<typename E > virtual int decaf::util::ListIterator< E >::nextIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next. (Returns list size if the list iterator is at the end of the list.)

Returns:

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

6.400.3.4 `template<typename E > virtual E decaf::util::ListIterator< E >::previous
() [pure virtual]`

Returns the previous element in the list. This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to `next` to go back and forth. (Note that alternating calls to `next` and `previous` will return the same element repeatedly.)

Returns:

the previous element in the list.

Exceptions:

NoSuchElementException - if the iteration has no previous element.

6.400.3.5 `template<typename E > virtual int decaf::util::ListIterator< E
>::previousIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to `previous`. (Returns -1 if the list iterator is at the beginning of the list.)

Returns:

the index of the element that would be returned by a subsequent call to `previous`, or -1 if list iterator is at beginning of list.

6.400.3.6 `template<typename E > virtual void
decaf::util::ListIterator< E >::set (const E & e) throw (
decaf::lang::exceptions::UnsupportedOperationException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException) [pure virtual]`

Replaces the last element returned by `next` or `previous` with the specified element (optional operation). This call can be made only if neither **ListIterator.remove** (p. 1833) nor **ListIterator.add** (p. 1991) have been called after the last call to `next` or `previous`.

Parameters:

e - the element with which to replace the last element returned by `next` or `previous`.

Exceptions:

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

IllegalStateException - if neither `next` nor `previous` have been called, or `remove` or `add` have been called after the last call to `next` or `previous`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ListIterator.h`

6.401 activemq::commands::LocalTransactionId Class Reference

#include <src/main/activemq/commands/LocalTransactionId.h> Inheritance diagram for activemq::commands::LocalTransactionId:

Public Types

- typedef decaf::lang::PointerComparator< LocalTransactionId > COMPARATOR

Public Member Functions

- LocalTransactionId ()
- LocalTransactionId (const LocalTransactionId &other)
- virtual ~LocalTransactionId ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual LocalTransactionId * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const DataStructure *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
Returns a string containing the information for this DataStructure (p. 1461) such as its type and value of its elements.
- virtual bool **equals** (const DataStructure *value) const
Compares the DataStructure (p. 1461) passed in to this one, and returns if they are equivalent.
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual const Pointer< ConnectionId > & **getConnectionId** () const
- virtual Pointer< ConnectionId > & **getConnectionId** ()
- virtual void **setConnectionId** (const Pointer< ConnectionId > &connectionId)
- virtual int **compareTo** (const LocalTransactionId &value) const
- virtual bool **equals** (const LocalTransactionId &value) const
- virtual bool **operator==** (const LocalTransactionId &value) const
- virtual bool **operator<** (const LocalTransactionId &value) const
- LocalTransactionId & **operator=** (const LocalTransactionId &other)

Static Public Attributes

- static const unsigned char **ID_LOCALTRANSACTIONID** = 111

Protected Attributes

- long long **value**
- **Pointer**< **ConnectionId** > **connectionId**

6.401.1 Member Typedef Documentation

6.401.1.1 **typedef** **decaf::lang::PointerComparator**<**LocalTransactionId**>
activemq::commands::LocalTransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 3217).

6.401.2 Constructor & Destructor Documentation

6.401.2.1 **activemq::commands::LocalTransactionId::LocalTransactionId** ()

6.401.2.2 **activemq::commands::LocalTransactionId::LocalTransactionId** (const
LocalTransactionId & *other*)

6.401.2.3 **virtual** **activemq::commands::LocalTransactionId::~~LocalTransactionId** ()
[**virtual**]

6.401.3 Member Function Documentation

6.401.3.1 **virtual** **LocalTransactionId*** **activemq::commands::LocalTransactionId::cloneDataStructure** () const
[**virtual**]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 3218).

6.401.3.2 **virtual** **int** **activemq::commands::LocalTransactionId::compareTo** (const
LocalTransactionId & *value*) const [**virtual**]

6.401.3.3 **virtual** **void** **activemq::commands::LocalTransactionId::copyDataStructure**
(**const** **DataStructure** * *src*) [**virtual**]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::TransactionId** (p. 3218).

6.401.3.4 `virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & value) const` [virtual]

6.401.3.5 `virtual bool activemq::commands::LocalTransactionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 3218).

6.401.3.6 `virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ()` [virtual]

6.401.3.7 `virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () const` [virtual]

6.401.3.8 `virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 3219).

- 6.401.3.9 `virtual long long activemq::commands::LocalTransactionId::getValue ()`
 `const [virtual]`
- 6.401.3.10 `virtual bool activemq::commands::LocalTransactionId::operator< (const`
 `LocalTransactionId & value) const [virtual]`
- 6.401.3.11 `LocalTransactionId& ac-`
 `tivemq::commands::LocalTransactionId::operator= (const`
 `LocalTransactionId & other)`
- 6.401.3.12 `virtual bool activemq::commands::LocalTransactionId::operator==`
 `(const LocalTransactionId & value) const [virtual]`
- 6.401.3.13 `virtual void activemq::commands::LocalTransactionId::setConnectionId`
 `(const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.401.3.14 `virtual void activemq::commands::LocalTransactionId::setValue (long`
 `long value) [virtual]`
- 6.401.3.15 `virtual std::string activemq::commands::LocalTransactionId::toString ()`
 `const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::TransactionId` (p. 3219).

6.401.4 Field Documentation

- 6.401.4.1 `Pointer<ConnectionId> ac-`
 `tivemq::commands::LocalTransactionId::connectionId`
 `[protected]`
- 6.401.4.2 `const unsigned char activemq::commands::LocalTransactionId::ID_ -`
 `LOCALTRANSACTIONID = 111 [static]`
- 6.401.4.3 `long long activemq::commands::LocalTransactionId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LocalTransactionId.h`

6.402 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1997).

#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.402.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1997). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.402.2 Constructor & Destructor Documentation

6.402.2.1 `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

6.402.2.2 `virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

6.402.3 Member Function Documentation

6.402.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.402.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.402.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3225).

6.402.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3225).

6.402.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3226).

6.402.3.6 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3226).

6.402.3.7 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3227).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LocalTransactionIdMarshaller.h**

6.403 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2001).

#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.403.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2001). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.403.2 Constructor & Destructor Documentation

6.403.2.1 `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

6.403.2.2 `virtual activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

6.403.3 Member Function Documentation

6.403.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.403.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.403.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3237).

6.403.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3237).

6.403.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3238).

6.403.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3238).

6.403.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3239).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h`

6.404 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2005).

#include <src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.404.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2005). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.404.2 Constructor & Destructor Documentation

6.404.2.1 `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

6.404.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

6.404.3 Member Function Documentation

6.404.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.404.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.404.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3229).

6.404.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3229).

6.404.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3230).

6.404.3.6 virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3230).

6.404.3.7 virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3231).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LocalTransactionIdMarshaller.h**

6.405 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.2009).

#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.405.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.2009). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.405.2 Constructor & Destructor Documentation

6.405.2.1 `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

6.405.2.2 `virtual activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

6.405.3 Member Function Documentation

6.405.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.405.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.405.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3221).

6.405.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3221).

6.405.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3222).

6.405.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3222).

6.405.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3223).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h`

6.406 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.2013).

#include <src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.406.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.2013). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.406.2 Constructor & Destructor Documentation

6.406.2.1 `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller()` [inline]

6.406.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller()` [inline, virtual]

6.406.3 Member Function Documentation

6.406.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.406.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.406.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3233).

6.406.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3233).

6.406.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3234).

6.406.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3234).

6.406.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3235).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h`

6.407 decaf::util::concurrent::locks::Lock Class Reference

Lock (p. 2017) implementations provide more extensive locking operations than can be obtained using synchronized statements.

#include <src/main/decaf/util/concurrent/locks/Lock.h> Inheritance diagram for decaf::util::concurrent::locks::Lock:

Public Member Functions

- virtual **~Lock** ()
- virtual void **lock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock.
- virtual void **lockInterruptibly** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock only if it is free at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Releases the lock.
- virtual **Condition** * **newCondition** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException)
*Returns a new **Condition** (p. 1118) instance that is bound to this **Lock** (p. 2017) instance.*

6.407.1 Detailed Description

Lock (p. 2017) implementations provide more extensive locking operations than can be obtained using synchronized statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 1118) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some locks may allow concurrent access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 2688).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor locks, and helps avoid many common programming errors involving locks, there are

occasions where you need to work with locks in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 2017) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of locks that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 2017) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...) { l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

Lock (p. 2017) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 2021)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 2019), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 2017) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 2017) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since:

1.0

6.407.2 Constructor & Destructor Documentation

6.407.2.1 `virtual decaf::util::concurrent::locks::Lock::~~Lock () [inline, virtual]`

6.407.3 Member Function Documentation

6.407.3.1 `virtual void decaf::util::concurrent::locks::Lock::lock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]`

Acquires the lock. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p.2017) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p.2017) implementation.

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p.2695).

6.407.3.2 virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException) [pure virtual]

Acquires the lock unless the current thread is interrupted. Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p.2017) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p.2017) implementation.

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p.2695).

6.407.3.3 virtual Condition* decaf::util::concurrent::locks::Lock::newCondition () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Returns a new **Condition** (p.1118) instance that is bound to this **Lock** (p.2017) instance. Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()**

(p. 1120) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

Implementation Considerations

The exact operation of the **Condition** (p. 1118) instance depends on the **Lock** (p. 2017) implementation and must be documented by that implementation.

Returns:

A new **Condition** (p. 1118) instance for this **Lock** (p. 2017) instance the caller must delete the returned **Condition** (p. 1118) object when done with it.

Exceptions:

RuntimeException if an error occurs while creating the **Condition** (p. 1118).

UnsupportedOperationException if this **Lock** (p. 2017) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2696).

6.407.3.4 virtual bool decaf::util::concurrent::locks::Lock::tryLock
(long long time, const TimeUnit & unit) throw
(decaf::lang::exceptions::RuntimeException, de-
caf::lang::exceptions::InterruptedException) [pure
virtual]

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted. If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or * The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p. 2017) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2017) implementation.

Parameters:

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns:

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2696).

6.407.3.5 virtual bool decaf::util::concurrent::locks::Lock::tryLock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Acquires the lock only if it is free at the time of invocation. Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

```
Lock (p. 2017) lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }
```

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

Returns:

true if the lock was acquired and false otherwise

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2697).

6.407.3.6 virtual void decaf::util::concurrent::locks::Lock::unlock () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Releases the lock. Implementation Considerations

A **Lock** (p. 2017) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 2017) implementation.

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

IllegalMonitorStateException if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2698).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/`**Lock.h**

6.408 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

Public Member Functions

- **Lock** (**Synchronizable** *object, const bool initiallyLocked=true)
Constructor - initializes the object member and locks the object if desired.
- virtual ~**Lock** ()
Destructor - Unlocks the object if it is locked.
- void **lock** ()
Locks the object.
- void **unlock** ()
Unlocks the object if it is already locked, otherwise a call to this method has no effect.
- bool **isLocked** () const
Indicates whether or not the object is locked.

6.408.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Since:

1.0

6.408.2 Constructor & Destructor Documentation

6.408.2.1 decaf::util::concurrent::Lock::Lock (**Synchronizable** * object, const bool *initiallyLocked* = true) [inline]

Constructor - initializes the object member and locks the object if desired.

Parameters:

object The sync object to control

initiallyLocked If true, the object will automatically be locked.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and lock().

6.408.2.2 virtual decaf::util::concurrent::Lock::~~Lock () [inline, virtual]

Destructor - Unlocks the object if it is locked.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::unlock().

6.408.3 Member Function Documentation**6.408.3.1 bool decaf::util::concurrent::Lock::isLocked () const [inline]**

Indicates whether or not the object is locked.

Returns:

true if the object is locked, otherwise false.

6.408.3.2 void decaf::util::concurrent::Lock::lock () [inline]

Locks the object.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::lock().

Referenced by Lock().

6.408.3.3 void decaf::util::concurrent::Lock::unlock () [inline]

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::unlock().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

6.409 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating locks and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

Public Member Functions

- `~LockSupport ()`

Static Public Member Functions

- static void **unpark** (decaf::lang::Thread *thread) throw ()
Makes available the permit for the given thread, if it was not already available.
- static void **park** () throw ()
Disables the current thread for thread scheduling purposes unless the permit is available.
- static void **parkNanos** (long long nanos) throw ()
Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.
- static void **parkUntil** (long long deadline) throw ()
Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

6.409.1 Detailed Description

Basic thread blocking primitives for creating locks and other synchronization classes. This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p.2827) class). A call to park will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to unpark makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods park and unpark provide efficient means of blocking and unblocking threads. Races between one thread invoking park and another thread trying to unpark it will preserve liveness, due to the permit. Additionally, park will return if the caller's thread was interrupted, and timeout versions are supported. The park method may also return at any other time, for "no reason", so in general must be invoked within a loop that rechecks conditions upon return. In this sense park serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an unpark to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The park method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither canProceed nor any other actions prior to the call to park entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of park could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:
AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;
public:
void lock() {
bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add( current );
// Block while not first in queue or cannot acquire lock while( waiters.peek() != current ||
!locked.compareAndSet( false, true ) ) {
LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting wasInter-
rupted = true; }
waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.interrupt(); }
void unlock() { locked.set( false ); LockSupport.unpark (p. 2027)( waiters.peek() ); } };
```

Since:

1.0

6.409.2 Constructor & Destructor Documentation

6.409.2.1 `decaf::util::concurrent::locks::LockSupport::~~LockSupport ()`

6.409.3 Member Function Documentation

6.409.3.1 `static void decaf::util::concurrent::locks::LockSupport::park () throw ()` [static]

Disables the current thread for thread scheduling purposes unless the permit is available. If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes unpark with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

6.409.3.2 `static void decaf::util::concurrent::locks::LockSupport::parkNanos (long long nanos) throw ()` [static]

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available. If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- * Some other thread invokes unpark with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

Parameters:

nanos the maximum number of nanoseconds to wait

6.409.3.3 static void decaf::util::concurrent::locks::LockSupport::parkUntil (long long *deadline*) throw () [static]

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available. If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes unpark with the current thread as the target; or * Some other thread interrupts the current thread; or * The specified deadline passes; or * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon return.

Parameters:

deadline the absolute time, in milliseconds from the Epoch, to wait until

6.409.3.4 static void decaf::util::concurrent::locks::LockSupport::unpark (decaf::lang::Thread * *thread*) throw () [static]

Makes available the permit for the given thread, if it was not already available. If the thread was blocked on park then it will unblock. Otherwise, its next call to park is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

Parameters:

thread the thread to unport, or NULL in which case the method has no effect.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**LockSupport.h**

6.410 decaf::util::logging::Logger Class Reference

```
#include <src/main/decaf/util/logging/Logger.h>
```

Public Member Functions

- **Logger** (const std::string &name, **Logger** *parent)
*Creates a new instance of the **Logger** (p. 2028) with the given name and assign it the given parent logger.*
- virtual ~**Logger** ()
- virtual const std::string & **getName** () const
*Gets the name of this **Logger** (p. 2028).*
- virtual void **addHandler** (**Handler** *handler) throw (lang::exceptions::IllegalArgumentException)
*Add a log **Handler** (p. 1709) to receive logging messages.*
- virtual void **removeHandler** (**Handler** *handler)
*Removes the specified **Handler** (p. 1709) and destroys it.*
- virtual void **setFilter** (**Filter** *filter)
Gets a vector containing all the handlers that this class has been assigned to use.
- virtual const **Filter** * **getFilter** () const
*Gets the **Filter** (p. 1630) object that this class is using.*
- virtual **Level** **getLevel** () const
*Get the log Level that has been specified for this **Logger** (p. 2028).*
- virtual void **setLevel** (**Level** level)
***Set** (p. 2905) the log level specifying which message levels will be logged by this logger.*
- virtual bool **getUseParentHandlers** () const
Discover whether or not this logger is sending its output to its parent logger.
- virtual void **setUseParentHandlers** (bool value)
*pecify whether or not this logger should send its output to it's parent **Logger** (p. 2028).*
- virtual void **entry** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Enter message.
- virtual void **exit** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Exit message.
- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)
Log a Debug Level Log.

- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)
Log a info Level Log.
- virtual void **warn** (const std::string &file, const int line, const std::string functionName, const std::string &message)
Log a warn Level Log.
- virtual void **error** (const std::string &file, const int line, const std::string functionName, const std::string &message)
Log a error Level Log.
- virtual void **fatal** (const std::string &file, const int line, const std::string functionName, const std::string &message)
Log a fatal Level Log.
- virtual bool **isLoggable** (**Level** level) const
Log a Throw Message.
- virtual void **log** (**LogRecord** &record)
*Log a **LogRecord** (p. 2050).*
- virtual void **log** (**Level** level, const std::string &message)
Log a message, with no arguments.
- virtual void **log** (**Level** level, const std::string &file, const int line, const std::string &message,...)
Log a message, with the list of params that is formatted into the message string.
- virtual void **log** (**Level** level, const std::string &file, const int line, const std::string &message, **lang::Exception** &ex)
Log a message, with associated Throwable information.

Static Public Member Functions

- static **Logger** * **getAnonymousLogger** ()
Creates an anonymous logger.
- static **Logger** * **getLogger** (const std::string &name)
Find or create a logger for a named subsystem.

6.410.1 Constructor & Destructor Documentation

6.410.1.1 decaf::util::logging::Logger::Logger (const std::string & name, Logger * parent)

Creates a new instance of the **Logger** (p. 2028) with the given name and assign it the given parent logger. The logger will be initially configured with a null Level and with useParentHandlers true.

Parameters:

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as java.net or javax.swing. It may be null for anonymous Loggers.

parent logger that is this one's parent

6.410.1.2 virtual decaf::util::logging::Logger::~~Logger () [virtual]

6.410.2 Member Function Documentation

6.410.2.1 virtual void decaf::util::logging::Logger::addHandler (Handler * *handler*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Add a log **Handler** (p.1709) to receive logging messages. By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p.2028) is configured with a set of Handlers that essentially act as default handlers for all loggers.

Parameters:

handler A Logging **Handler** (p.1709)

Exceptions:

IllegalArgumentException

6.410.2.2 virtual void decaf::util::logging::Logger::debug (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a Debug Level Log. If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p.1709) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

6.410.2.3 virtual void decaf::util::logging::Logger::entry (const std::string & *blockName*, const std::string & *file*, const int *line*) [virtual]

Logs an Block Enter message. This is a convenience method that is used to tag a block enter, a log record with the class name function name and the string Entering is logged at the DEBUG log level.

Parameters:

blockName source block name

file source file name

line source line name

6.410.2.4 virtual void decaf::util::logging::Logger::error (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a error Level Log. If the logger is currently enabled for the error message level then the given message is forwarded to all the registered output **Handler** (p.1709) objects.

Parameters:

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log

6.410.2.5 virtual void decaf::util::logging::Logger::exit (const std::string & *blockName*, const std::string & *file*, const int *line*) [virtual]

Logs an Block Exit message. This is a convenience method that is used to tag a block exit, a log record with the class name function name and the string Exiting is logged at the DEBUG log level.

Parameters:

blockName source block name
file source file name
line source line name

6.410.2.6 virtual void decaf::util::logging::Logger::fatal (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a fatal Level Log. If the logger is currently enabled for the fatal message level then the given message is forwarded to all the registered output **Handler** (p.1709) objects.

Parameters:

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log

6.410.2.7 static Logger* decaf::util::logging::Logger::getAnonymousLogger () [static]

Creates an anonymous logger. The newly created **Logger** (p.2028) is not registered in the **Log-Manager** (p.2045) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger (") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

Returns:

Newly created anonymous logger

6.410.2.8 `virtual const Filter* decaf::util::logging::Logger::getFilter () const`
[inline, virtual]

Gets the **Filter** (p. 1630) object that this class is using.

Returns:

the **Filter** (p. 1630) in use, can be null

6.410.2.9 `virtual Level decaf::util::logging::Logger::getLevel () const` [inline, virtual]

Get the log Level that has been specified for this **Logger** (p. 2028). The result may be the Null level, which means that this logger's effective level will be inherited from its parent.

Returns:

the level that is currently set

6.410.2.10 `static Logger* decaf::util::logging::Logger::getLogger (const std::string & name)` [static]

Find or create a logger for a named subsystem. If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 2045) and it will be configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 2045) global namespace.

Parameters:

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as `cms` or `activemq.core.ActiveMQConnection` (p. 233)

Returns:

a suitable logger.

6.410.2.11 `virtual const std::string& decaf::util::logging::Logger::getName () const`
[inline, virtual]

Gets the name of this **Logger** (p. 2028).

Returns:

logger name

6.410.2.12 `virtual bool decaf::util::logging::Logger::getUseParentHandlers () const`
[inline, virtual]

Discover whether or not this logger is sending its output to its parent logger.

Returns:

true if using Parent Handlers

6.410.2.13 `virtual void decaf::util::logging::Logger::info (const std::string & file,
const int line, const std::string functionName, const std::string &
message)` [virtual]

Log a info Level Log. If the logger is currently enabled for the info message level then the given message is forwarded to all the registered output **Handler** (p.1709) objects.

Parameters:

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log

6.410.2.14 `virtual bool decaf::util::logging::Logger::isLoggable (Level level) const`
[virtual]

Log a Throw Message. If the logger is currently enabled for the Throwing message level then the given message is forwarded to all the registered output **Handler** (p.1709) objects.

Parameters:

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log
`virtual void throwing(const std::string& file, const int line,
const std::string functionName, const std::string& message);` Check if a message of the given level would actually be logged by this logger. This check is based on the Loggers effective level, which may be inherited from its parent.
level - a message logging level

Returns:

true if the given message level is currently being logged.

6.410.2.15 `virtual void decaf::util::logging::Logger::log (Level level, const
std::string & file, const int line, const std::string & message,
lang::Exception & ex)` [virtual]

Log a message, with associated Throwable information. If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p.2050) which is

forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 2050) thrown property, rather than the **LogRecord** (p. 2050) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 2050) message property.

Parameters:

level the Level to log at.
file File that the message was logged in.
line the line number where the message was logged at.
message the message to log.
ex the Exception to log

6.410.2.16 `virtual void decaf::util::logging::Logger::log (Level level, const std::string & file, const int line, const std::string & message, ...)`
 [virtual]

Log a message, with the list of params that is formatted into the message string. If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1709) objects

Parameters:

level the Level to log at
file the message to log
line the line in the file
 ... variable length argument to format the message string.

6.410.2.17 `virtual void decaf::util::logging::Logger::log (Level level, const std::string & message)` [virtual]

Log a message, with no arguments. If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1709) objects

Parameters:

level the Level to log at
message the message to log

6.410.2.18 `virtual void decaf::util::logging::Logger::log (LogRecord & record)`
 [virtual]

Log a **LogRecord** (p. 2050). All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

Parameters:

record - the **LogRecord** (p. 2050) to be published

6.410.2.19 virtual void decaf::util::logging::Logger::removeHandler (Handler * *handler*) [virtual]

Removes the specified **Handler** (p. 1709) and destroys it. Returns silently if the given **Handler** (p. 1709) is not found.

Parameters:

handler The **Handler** (p. 1709) to remove

6.410.2.20 virtual void decaf::util::logging::Logger::setFilter (Filter * *filter*) [virtual]

Gets a vector containing all the handlers that this class has been assigned to use.

Returns:

a list of handlers that are used by this logger **Set** (p. 2905) a filter to control output on this **Logger** (p. 2028).

After passing the initial "level" check, the **Logger** (p. 2028) will call this **Filter** (p. 1630) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

Parameters:

filter to use, can be null

6.410.2.21 virtual void decaf::util::logging::Logger::setLevel (Level *level*) [inline, virtual]

Set (p. 2905) the log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded. The level value Level.OFF can be used to turn off logging.

If the new level is the Null Level, it means that this node should inherit its level from its nearest ancestor with a specific (non-null) level value.

Parameters:

level new Level value

6.410.2.22 virtual void decaf::util::logging::Logger::setUseParentHandlers (bool *value*) [inline, virtual]

pecify whether or not this logger should send its output to it's parent **Logger** (p. 2028). This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

Parameters:

value True is output is to be written to the parent

6.410.2.23 `virtual void decaf::util::logging::Logger::warn (const std::string & file,
const int line, const std::string functionName, const std::string &
message)` [virtual]

Log a warn Level Log. If the logger is currently enabled for the warn message level then the given message is forwarded to all the registered output **Handler** (p. 1709) objects.

Parameters:

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Logger.h`

6.411 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

Public Member Functions

- **LoggerHierarchy** ()
- virtual **~LoggerHierarchy** ()

6.411.1 Constructor & Destructor Documentation

6.411.1.1 decaf::util::logging::LoggerHierarchy::LoggerHierarchy ()

6.411.1.2 virtual decaf::util::logging::LoggerHierarchy::~~LoggerHierarchy ()
[virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LoggerHierarchy.h`

6.412 activemq::io::LoggingInputStream Class Reference

#include <src/main/activemq/io/LoggingInputStream.h> Inheritance diagram for activemq::io::LoggingInputStream:

Public Member Functions

- **LoggingInputStream** (decaf::io::InputStream *inputStream, bool own=false)
Creates a DataInputStream that uses the specified underlying InputStream.
- virtual ~**LoggingInputStream** ()
- virtual int **read** () throw (decaf::io::IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.

6.412.1 Constructor & Destructor Documentation

6.412.1.1 activemq::io::LoggingInputStream::LoggingInputStream (decaf::io::InputStream * inputStream, bool own = false)

Creates a DataInputStream that uses the specified underlying InputStream.

Parameters:

- inputStream* the InputStream instance to wrap.
- own* indicates if this class owns the wrapped string defaults to false.

6.412.1.2 virtual activemq::io::LoggingInputStream::~~LoggingInputStream () [virtual]

6.412.2 Member Function Documentation

6.412.2.1 virtual int activemq::io::LoggingInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer. Blocks until the requested number of bytes are available.

Parameters:

- buffer* (out) the target buffer.
- offset* the position in the buffer to start at
- bufferSize* the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF is detected

Exceptions:

IOException thrown if an error occurs.

NullPointerException if buffer is null

Reimplemented from **decaf::io::FilterInputStream** (p. 1635).

6.412.2.2 virtual int activemq::io::LoggingInputStream::read () throw (decaf::io::IOException) [virtual]

Reads a single byte from the buffer. Blocks until data is available.

Returns:

The next byte.

Exceptions:

IOException thrown if an error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1636).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`

6.413 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

#include <src/main/activemq/io/LoggingOutputStream.h> Inheritance diagram for activemq::io::LoggingOutputStream:

Public Member Functions

- **LoggingOutputStream** (OutputStream *next, bool own=false)
Constructor.
- virtual ~**LoggingOutputStream** ()
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.

6.413.1 Detailed Description

OutputStream filter that just logs the data being written.

6.413.2 Constructor & Destructor Documentation

6.413.2.1 activemq::io::LoggingOutputStream::LoggingOutputStream (OutputStream * next, bool own = false)

Constructor.

Parameters:

- next* The OutputStream to wrap an write logs to.
- own* If true, this object will control the lifetime of the output stream that it encapsulates.

6.413.2.2 virtual activemq::io::LoggingOutputStream::~~LoggingOutputStream () [virtual]

6.413.3 Member Function Documentation

6.413.3.1 virtual void activemq::io::LoggingOutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.
offset the position in the buffer to start at
len The number of bytes from the buffer to be written.

Exceptions:

IOException thrown if an error occurs.
NullPointerException if buffer is null.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1645).

**6.413.3.2 virtual void activemq::io::LoggingOutputStream::write (unsigned char *c*)
throw (decaf::io::IOException) [virtual]**

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1646).

The documentation for this class was generated from the following file:

- src/main/activemq/io/**LoggingOutputStream.h**

6.414 activemq::transport::logging::LoggingTransport Class Reference

A transport filter that logs commands as they are sent/received.

#include <src/main/activemq/transport/logging/LoggingTransport.h> Inheritance diagram for activemq::transport::logging::LoggingTransport:

Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &next)

Constructor.

- virtual ~**LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

Event handler for the receipt of a command.

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Not supported by this class - throws an exception.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Not supported by this class - throws an exception.

6.414.1 Detailed Description

A transport filter that logs commands as they are sent/received.

6.414.2 Constructor & Destructor Documentation

6.414.2.1 activemq::transport::logging::LoggingTransport::LoggingTransport (const **Pointer**< **Transport** > & next)

Constructor.

Parameters:

next - the next **Transport** (p. 3273) in the chain

6.414.2.2 virtual
activemq::transport::logging::LoggingTransport::~~LoggingTransport ()
[inline, virtual]

6.414.3 Member Function Documentation

6.414.3.1 virtual void activemq::transport::logging::LoggingTransport::onCommand
(const Pointer< Command > & *command*) [virtual]

Event handler for the receipt of a command.

Parameters:

command - the received command object.

Reimplemented from **activemq::transport::TransportFilter** (p. 3285).

6.414.3.2 virtual void activemq::transport::logging::LoggingTransport::oneway
(const Pointer< Command > & *command*) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3285).

6.414.3.3 virtual Pointer<Response> ac-
tivemq::transport::logging::LoggingTransport::request
(const Pointer< Command > & *command*, un-
signed int *timeout*) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported by this class - throws an exception.

Parameters:

command the command that is sent as a request

timeout the time to wait for a response.

Exceptions:

UnsupportedOperationException.

Reimplemented from **activemq::transport::TransportFilter** (p. 3286).

6.414.3.4 `virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters:

command the command that is sent as a request

Exceptions:

UnsupportedOperationException.

Reimplemented from `activemq::transport::TransportFilter` (p. 3287).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/logging/LoggingTransport.h`

6.415 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p.2045) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```

Public Member Functions

- virtual **~LogManager** ()
- virtual void **setProperties** (const **util::Properties** &properties)
*Sets the **Properties** (p.2657) this **LogManager** (p.2045) should use to configure its loggers.*
- virtual const **util::Properties** & **getProperties** () const
*Gets a reference to the Logging **Properties** (p.2657) used by this logger.*
- virtual std::string **getProperty** (const std::string &name)
*Gets the value of a named property of this **LogManager** (p.2045).*
- virtual void **addPropertyChangeListener** (PropertyChangeListener *listener)
*Adds a change listener for **LogManager** (p.2045) **Properties** (p.2657), adding the same instance of a change event listener does nothing.*
- virtual void **removePropertyChangeListener** (PropertyChangeListener *listener)
*Removes a properties change listener from the **LogManager** (p.2045).*
- virtual **Logger** * **getLogger** (const std::string &name)
*Retrieves or creates a new **Logger** (p.2028) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- virtual int **getLoggerNames** (const std::vector< std::string > &names)
*Gets a list of known **Logger** (p.2028) Names from this Manager.*

Static Public Member Functions

- static **LogManager** * **getInstance** ()
Get the singleton instance.
- static void **returnInstance** ()
Returns a Checked out instance of this Manager.
- static void **destroy** ()
*Forcefully Delete the Instance of this **LogManager** (p.2045) even if there are outstanding references.*

Protected Member Functions

- **LogManager** ()
Constructor, hidden to protect against direct instantiation.
- **LogManager** (const **LogManager** &manager)
Copy Constructo.
- void **operator=** (const **LogManager** &manager)
Assignment operator.

6.415.1 Detailed Description

There is a single global **LogManager** (p. 2045) object that is used to maintain a set of shared state about Loggers and log services. This **LogManager** (p. 2045) object:

* Manages a hierarchical namespace of **Logger** (p. 2028) objects. All named Loggers are stored in this namespace. * Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global **LogManager** (p. 2045) object can be retrieved using `LogManager.getLogManager()`. The **LogManager** (p. 2045) object is created during class initialization and cannot subsequently be changed.

TODO By default, the **LogManager** (p. 2045) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the **LogManager** (p. 2045) uses two optional system properties that allow more control over reading the initial configuration:

* "decaf.logger.config.class" * "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to `JNI_CreateJavaVM`.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use `readConfiguration(InputStream)` to define properties in the **LogManager** (p. 2045).

If "java.util.logging.config.class" property is not set, then the "java.util.logging.config.file" system property can be used to specify a properties file (in `java.util.Properties` format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 2045) will read its initial configuration from a properties file "lib/logging.properties" in the JRE directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. ***TODO***

The global logging properties may include:

* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 2028) (the **Logger** (p. 2028) named ""). Each class name must be for a **Handler** (p. 1709) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.handlers".

This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1709) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1709) needs to be configured for this logger otherwise no logging messages are delivered. * A property "config". This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and "a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 2045) object are multi-thread safe.

6.415.2 Constructor & Destructor Documentation

6.415.2.1 virtual decaf::util::logging::LogManager::~~LogManager () [virtual]

6.415.2.2 decaf::util::logging::LogManager::LogManager () [inline, protected]

Constructor, hidden to protect against direct instantiation.

6.415.2.3 decaf::util::logging::LogManager::LogManager (const LogManager & *manager*) [protected]

Copy Constructo.

Parameters:

manager the Manager to copy

6.415.3 Member Function Documentation

6.415.3.1 virtual void decaf::util::logging::LogManager::addPropertyChangeListener (PropertyChangeListener * *listener*) [virtual]

Adds a change listener for **LogManager** (p. 2045) **Properties** (p. 2657), adding the same instance of a change event listener does nothing.

Parameters:

listener a PropertyChangeListener

6.415.3.2 static void decaf::util::logging::LogManager::destroy () [static]

Forcefully Delete the Instance of this **LogManager** (p. 2045) even if there are outstanding references.

6.415.3.3 static LogManager* decaf::util::logging::LogManager::getInstance () [static]

Get the singleton instance.

Returns:

Pointer to an instance of the Log Manager

6.415.3.4 virtual Logger* decaf::util::logging::LogManager::getLogger (const std::string & name) [virtual]

Retrieves or creates a new **Logger** (p. 2028) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

Parameters:

name The name of the **Logger** (p. 2028).

6.415.3.5 virtual int decaf::util::logging::LogManager::getLoggerNames (const std::vector< std::string > & names) [virtual]

Gets a list of known **Logger** (p. 2028) Names from this Manager.

Parameters:

names STL Vector to hold string logger names

Returns:

names count of how many loggers were inserted

6.415.3.6 virtual const util::Properties& decaf::util::logging::LogManager::getProperties () const [inline, virtual]

Gets a reference to the Logging **Properties** (p. 2657) used by this logger.

Returns:

The **Logger** (p. 2028) **Properties** (p. 2657) Pointer

6.415.3.7 virtual std::string decaf::util::logging::LogManager::getProperty (const std::string & name) [inline, virtual]

Gets the value of a named property of this **LogManager** (p. 2045).

Parameters:

name of the Property to retrieve

Returns:

the value of the property

References decaf::util::Properties::getProperty().

6.415.3.8 void decaf::util::logging::LogManager::operator= (const LogManager & *manager*) [protected]

Assignment operator.

Parameters:

manager the manager to assign from

6.415.3.9 virtual void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener * *listener*) [virtual]

Removes a properties change listener from the **LogManager** (p. 2045).

Parameters:

listener a PropertyChangeListener

6.415.3.10 static void decaf::util::logging::LogManager::returnInstance () [static]

Returns a Checked out instance of this Manager.

6.415.3.11 virtual void decaf::util::logging::LogManager::setProperties (const util::Properties & *properties*) [virtual]

Sets the **Properties** (p. 2657) this **LogManager** (p. 2045) should use to configure its loggers. Once set a properties change event is fired.

Parameters:

properties Pointer to read the configuration from

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogManager.h**

6.416 decaf::util::logging::LogRecord Class Reference

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

Public Member Functions

- **LogRecord** ()
- virtual **~LogRecord** ()
- **Level** **getLevel** () const
Get Level of this log record.
- void **setLevel** (**Level** value)
Set (p. 2905) the Level of this Log Record.
- const std::string & **getLoggerName** () const
Gets the Source Logger's Name.
- void **setLoggerName** (const std::string &loggerName)
Sets the Source Logger's Name.
- const std::string & **getSourceFile** () const
Gets the Source Log File name.
- void **setSourceFile** (const std::string &sourceFile)
Sets the Source Log File Name.
- unsigned long **getSourceLine** () const
Gets the Source Log line number.
- void **setSourceLine** (long sourceLine)
Sets the Source Log line number.
- const std::string & **getMessage** () const
Gets the Message to be Logged.
- void **setMessage** (const std::string &message)
Sets the Message to be Logged.
- const std::string & **getSourceFunction** () const
Gets the name of the function where this log was logged.
- void **setSourceFunction** (const std::string &functionName)
Sets the name of the function where this log was logged.
- unsigned long **getTimestamp** () const
Gets the time in mills that this message was logged.
- void **setTimestamp** (long timeStamp)
Sets the time in mills that this message was logged.

- unsigned long **getTreadId** () const
Gets the Thread Id where this Log was created.
- void **setTreadId** (long threadId)
Sets the Thread Id where this Log was created.

6.416.1 Constructor & Destructor Documentation

6.416.1.1 `decaf::util::logging::LogRecord::LogRecord ()` [inline]

6.416.1.2 `virtual decaf::util::logging::LogRecord::~~LogRecord ()` [inline, virtual]

6.416.2 Member Function Documentation

6.416.2.1 `Level decaf::util::logging::LogRecord::getLevel () const` [inline]

Get Level of this log record.

Returns:

Level enumeration value.

6.416.2.2 `const std::string& decaf::util::logging::LogRecord::getLoggerName () const` [inline]

Gets the Source Logger's Name.

Returns:

the source loggers name

6.416.2.3 `const std::string& decaf::util::logging::LogRecord::getMessage () const` [inline]

Gets the Message to be Logged.

Returns:

the source logger's message

Referenced by `decaf::util::logging::SimpleFormatter::formatMessage()`.

6.416.2.4 `const std::string& decaf::util::logging::LogRecord::getSourceFile () const` [inline]

Gets the Source Log File name.

Returns:

the source loggers name

6.416.2.5 `const std::string& decaf::util::logging::LogRecord::getSourceFunction () const [inline]`

Gets the name of the function where this log was logged.

Returns:

the source logger's message

6.416.2.6 `unsigned long decaf::util::logging::LogRecord::getSourceLine () const [inline]`

Gets the Source Log line number.

Returns:

the source loggers line number

6.416.2.7 `unsigned long decaf::util::logging::LogRecord::getTimestamp () const [inline]`

Gets the time in mills that this message was logged.

Returns:

UTC time in milliseconds

6.416.2.8 `unsigned long decaf::util::logging::LogRecord::getTreadId () const [inline]`

Gets the Thread Id where this Log was created.

Returns:

the source loggers line number

6.416.2.9 `void decaf::util::logging::LogRecord::setLevel (Level value) [inline]`

Set (p. 2905) the Level of this Log Record.

Parameters:

value Level Enumeration Value

6.416.2.10 `void decaf::util::logging::LogRecord::setLoggerName (const std::string & loggerName) [inline]`

Sets the Source Logger's Name.

Parameters:

loggerName the source loggers name

6.416.2.11 `void decaf::util::logging::LogRecord::setMessage (const std::string & message)` [inline]

Sets the Message to be Logged.

Parameters:

message the source loggers message

6.416.2.12 `void decaf::util::logging::LogRecord::setSourceFile (const std::string & sourceFile)` [inline]

Sets the Source Log File Name.

Parameters:

sourceFile the source loggers name

6.416.2.13 `void decaf::util::logging::LogRecord::setSourceFunction (const std::string & functionName)` [inline]

Sets the name of the function where this log was logged.

Parameters:

functionName the source of the log

6.416.2.14 `void decaf::util::logging::LogRecord::setSourceLine (long sourceLine)` [inline]

Sets the Source Log line number.

Parameters:

sourceLine the source logger's line number

6.416.2.15 `void decaf::util::logging::LogRecord::setTimestamp (long timeStamp)` [inline]

Sets the time in mills that this message was logged.

Parameters:

timeStamp UTC Time in Milliseconds.

6.416.2.16 `void decaf::util::logging::LogRecord::setTreadId (long threadId)` [inline]

Sets the Thread Id where this Log was created.

Parameters:

threadId the source logger's line number

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LogRecord.h`

6.417 decaf::util::logging::LogWriter Class Reference

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

Public Member Functions

- **LogWriter** ()
- virtual **~LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)
Writes a message to the output destination.
- virtual void **log** (const std::string &message)
Writes a message to the output destination.

Static Public Member Functions

- static **LogWriter** & **getInstance** ()
Get the singleton instance.
- static void **returnInstance** ()
Returns a Checked out instance of this Writer.
- static void **destroy** ()
*Forcefully Delete the Instance of this **LogWriter** (p. 2055) even if there are outstanding references.*

6.417.1 Constructor & Destructor Documentation

6.417.1.1 decaf::util::logging::LogWriter::LogWriter ()

6.417.1.2 virtual decaf::util::logging::LogWriter::~~LogWriter () [virtual]

6.417.2 Member Function Documentation

6.417.2.1 static void decaf::util::logging::LogWriter::destroy () [static]

Forcefully Delete the Instance of this **LogWriter** (p. 2055) even if there are outstanding references.

6.417.2.2 static **LogWriter**& decaf::util::logging::LogWriter::getInstance ()
[static]

Get the singleton instance.

6.417.2.3 **virtual void decaf::util::logging::LogWriter::log** (const std::string & *message*) [virtual]

Writes a message to the output destination.

Parameters:

message

6.417.2.4 **virtual void decaf::util::logging::LogWriter::log** (const std::string & *file*,
const int *line*, const std::string & *prefix*, const std::string & *message*)
[virtual]

Writes a message to the output destination.

Parameters:

file

line

prefix

message

6.417.2.5 **static void decaf::util::logging::LogWriter::returnInstance** () [static]

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogWriter.h**

6.418 decaf::lang::Long Class Reference

#include <src/main/decaf/lang/Long.h> Inheritance diagram for decaf::lang::Long:

Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual **~Long** ()
- virtual int **compareTo** (const **Long** &l) const
*Compares this **Long** (p. 2057) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Long** &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const long long &l) const
*Compares this **Long** (p. 2057) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const long long &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static int **bitCount** (long long value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static **Long decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Long** (p. 2057).*
- static long long **highestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
- static long long **lowestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (long long value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.
- static int **numberOfTrailingZeros** (long long value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.
- static long long **parseLong** (const std::string &value) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal long.
- static long long **parseLong** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Long** (p. 2057) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*
- static long long **reverseBytes** (long long value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.
- static long long **reverse** (long long value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.
- static long long **rotateLeft** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.
- static long long **rotateRight** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.
- static int **signum** (long long value)

Returns the signum function of the specified value.

- static std::string **toString** (long long value)

Converts the long to a String representation.

- static std::string **toString** (long long value, int radix)
- static std::string **toHexString** (long long value)

Returns a string representation of the integer argument as an unsigned integer in base 16.

- static std::string **toOctalString** (long long value)

Returns a string representation of the long long argument as an unsigned long long in base 8.

- static std::string **toBinaryString** (long long value)

Returns a string representation of the long long argument as an unsigned long long in base 2.

- static **Long valueOf** (long long value)

*Returns a **Long** (p. 2057) instance representing the specified int value.*

- static **Long valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a **Long** (p. 2057) object holding the value given by the specified std::string.*

- static **Long valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

*Returns a **Long** (p. 2057) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 64

The size in bits of the primitive long long type.

- static const long long **MAX_VALUE** = (long long)0x7FFFFFFFFFFFFFFFFFLL

The maximum value that the primitive type can hold.

- static const long long **MIN_VALUE** = (long long)0x8000000000000000LL

The minimum value that the primitive type can hold.

6.418.1 Constructor & Destructor Documentation

6.418.1.1 decaf::lang::Long::Long (long long value)

Parameters:

value - the primitive long long to wrap

6.418.1.2 `decaf::lang::Long::Long (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters:

value - the long long formatted string to wrap

Exceptions:

NumberFormatException

6.418.1.3 `virtual decaf::lang::Long::~~Long () [inline, virtual]`

6.418.2 Member Function Documentation

6.418.2.1 `static int decaf::lang::Long::bitCount (long long value) [static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the population count.

Parameters:

value - the long long to count

Returns:

the number of one-bits in the two's complement binary representation of the specified long long value.

6.418.2.2 `virtual unsigned char decaf::lang::Long::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2432).

6.418.2.3 `virtual int decaf::lang::Long::compareTo (const long long & l) const [virtual]`

Compares this **Long** (p. 2057) instance with another.

Parameters:

l - the **Integer** (p. 1762) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< long long >` (p. 1083).

6.418.2.4 virtual int decaf::lang::Long::compareTo (const Long & *l*) const
[virtual]

Compares this **Long** (p. 2057) instance with another.

Parameters:

l - the **Long** (p. 2057) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Long** > (p. 1083).

6.418.2.5 static Long decaf::lang::Long::decode (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Long** (p. 2057). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Long** (p. 2057) object containing the decoded value

Exceptions:

NumberFormatException if the string is not formatted correctly.

6.418.2.6 virtual double decaf::lang::Long::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.418.2.7 bool decaf::lang::Long::equals (const long long & *l*) const [inline, virtual]**Parameters:**

l - the **Long** (p. 2057) object to compare against.

6.418.2.12 `virtual long long decaf::lang::Long::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.418.2.13 `static long long decaf::lang::Long::lowestOneBit (long long value) [static]`

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the long long to be inspected

Returns:

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.418.2.14 `static int decaf::lang::Long::numberOfLeadingZeros (long long value) [static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value. Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

$\ast \text{floor}(\log_2(x)) = 63 - \text{numberOfLeadingZeros}(x) \ast \text{ceil}(\log_2(x)) = 64 - \text{numberOfLeadingZeros}(x - 1)$

Parameters:

value - the long long to be inspected

Returns:

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.418.2.15 `static int decaf::lang::Long::numberOfTrailingZeros (long long value) [static]`

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value. Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.418.2.16 `virtual bool decaf::lang::Long::operator< (const long long & l) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1084).

6.418.2.17 `virtual bool decaf::lang::Long::operator< (const Long & l) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< Long >` (p. 1084).

6.418.2.18 `virtual bool decaf::lang::Long::operator== (const long long & l) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1085).

6.418.2.19 **virtual bool decaf::lang::Long::operator== (const Long & l) const**
 [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Long >** (p. 1085).

6.418.2.20 **static long long decaf::lang::Long::parseLong (const std::string & value,**
 int radix) throw (exceptions::NumberFormatException) [static]

Returns a **Long** (p. 2057) object holding the value extracted from the specified string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 2057) object that represents the long long value specified by the string.

Parameters:

value - String to parse

radix - the base encoding of the string

Returns:

long long value

Exceptions:

NumberFormatException on invalid string value

6.418.2.21 **static long long decaf::lang::Long::parseLong (const std::string & value)**
 throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal long. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

Parameters:

value - String to parse

Returns:

long long value

Exceptions:

NumberFormatException on invalid string value

6.418.2.22 static long long decaf::lang::Long::reverse (long long *value*) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

Parameters:

value - the value whose bits are to be reversed

Returns:

the reversed bits long long.

6.418.2.23 static long long decaf::lang::Long::reverseBytes (long long *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

Parameters:

value - the long long whose bytes we are to reverse

Returns:

the reversed long long.

6.418.2.24 static long long decaf::lang::Long::rotateLeft (long long *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters:

value - the long long to be inspected

distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

6.418.2.25 `static long long decaf::lang::Long::rotateRight (long long value, int distance)` [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

Parameters:

value - the long long to be inspected
distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

6.418.2.26 `virtual short decaf::lang::Long::shortValue () const` [inline, virtual]

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2434).

6.418.2.27 `static int decaf::lang::Long::signum (long long value)` [static]

Returns the signum function of the specified value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters:

value - the long long to be inspected

Returns:

the signum function of the specified long long value.

6.418.2.28 `static std::string decaf::lang::Long::toBinaryString (long long value)` [static]

Returns a string representation of the long long argument as an unsigned long long in base 2. The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

Parameters:

value - the long long to be translated to a binary string

Returns:

the unsigned long long value as a binary string

6.418.2.29 static std::string decaf::lang::Long::toHexString (long long *value*)
[static]

Returns a string representation of the integer argument as an unsigned integer in base 16. The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters:

value - the long long to be translated to an Octal string

Returns:

the unsigned long long value as a Octal string

6.418.2.30 static std::string decaf::lang::Long::toOctalString (long long *value*)
[static]

Returns a string representation of the long long argument as an unsigned long long in base 8. The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters:

value - the long long to be translated to an Octal string

Returns:

the unsigned long long value as a Octal string


```
6.418.2.31 static std::string decaf::lang::Long::toString (long long value, int radix)
[static]
```

6.418.2.32 `static std::string decaf::lang::Long::toString (long long value) [static]`

Converts the long to a String representation.

Parameters:

value The long to convert to a std::string.

Returns:

string representation

6.418.2.33 std::string decaf::lang::Long::toString () const

Returns:

this **Long** (p. 2057) Object as a String Representation

```
6.418.2.34 static Long decaf::lang::Long::valueOf (const std::string & value, int
radix) throw ( exceptions::NumberFormatException ) [static]
```

Returns a **Long** (p. 2057) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 2057) object that represents the long long value specified by the string.

Parameters:

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Long** (p. 2057) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid long long.

```
6.418.2.35 static Long defaf::lang::Long::valueOf (const std::string & value) throw
( exceptions::NumberFormatException ) [static]
```

Returns a **Long** (p.2057) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the `parseLong(std::string)` method. The result is a **Integer** (p.1762) object that represents the long long value specified by the string.

Parameters:

value - std::string to parse as base 10

Returns:

new **Long** (p. 2057) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal long long.

6.418.2.36 `static Long decaf::lang::Long::valueOf (long long value) [inline, static]`

Returns a **Long** (p. 2057) instance representing the specified int value.

Parameters:

value - the long long to wrap

Returns:

the new **Integer** (p. 1762) object wrapping value.

6.418.3 Field Documentation

6.418.3.1 `const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFFLL [static]`

The maximum value that the primitive type can hold.

6.418.3.2 `const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL [static]`

The minimum value that the primitive type can hold.

6.418.3.3 `const int decaf::lang::Long::SIZE = 64 [static]`

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Long.h`

6.419 decaf::internal::nio::LongArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/LongArrayBuffer.h> Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

Public Member Functions

- **LongArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1744) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **LongArrayBuffer** (long long *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **LongArrayBuffer** (p. 2071) object that wraps the given array.*
- **LongArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset.*
- **LongArrayBuffer** (const LongArrayBuffer &other)
*Create a **LongArrayBuffer** (p. 2071) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~**LongArrayBuffer** ()
- virtual long long * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the long long array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual LongBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only long long buffer that shares this buffer's content.
- virtual LongBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual LongBuffer * **duplicate** ()
Creates a new long long buffer that shares this buffer's content.
- virtual long long **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual long long **get** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible long long array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual LongBuffer & **put** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual LongBuffer & **put** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual LongBuffer * **slice** () const
Creates a new LongBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteArrayBuffer** (p. 870) as Read-Only.*

6.419.1 Constructor & Destructor Documentation

6.419.1.1 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **IntArrayBuffer** (p. 1744) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.419.1.2 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **LongArrayBuffer** (p. 2071) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.419.1.3 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset. The capacity and limit of the new **LongArrayBuffer** (p. 2071) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 908) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.419.1.4 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const LongArrayBuffer & other)

Create a **LongArrayBuffer** (p. 2071) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.

Parameters:

other - the **LongArrayBuffer** (p. 2071) this one is to mirror.

6.419.1.5 virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer () [virtual]

6.419.2 Member Function Documentation

6.419.2.1 virtual long long* decaf::internal::nio::LongArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the long long array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2082).

6.419.2.2 `virtual std::size_t decaf::internal::nio::LongArrayBuffer::arrayOffset
() throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2082).

6.419.2.3 `virtual LongBuffer* de-
caf::internal::nio::LongArrayBuffer::asReadOnlyBuffer ()
const [virtual]`

Creates a new, read-only long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 2082).

6.419.2.4 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 807) is copied

to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index `limit()` (p. 807) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this LongBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::LongBuffer` (p. 2083).

6.419.2.5 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::duplicate ()`
[virtual]

Creates a new long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new long long Buffer which the caller owns.

Implements `decaf::nio::LongBuffer` (p. 2083).

6.419.2.6 `virtual long long decaf::internal::nio::LongArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the long long is to be read

Returns:

the long long that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements `decaf::nio::LongBuffer` (p. 2085).

6.419.2.7 `virtual long long decaf::internal::nio::LongArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the long long at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements `decaf::nio::LongBuffer` (p. 2085).

6.419.2.8 `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible long long array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::LongBuffer` (p. 2085).

6.419.2.9 `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements `decaf::nio::Buffer` (p. 806).

6.419.2.10 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2086).

6.419.2.11 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put
(long long value) throw (decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException)` [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2086).

6.419.2.12 `virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly (bool
value)` [inline, protected, virtual]

Sets this **ByteBuffer** (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.419.2.13 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice () const` [virtual]

Creates a new **LongBuffer** whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **LongBuffer** which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2088).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/LongArrayBuffer.h`

6.420 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:.

```
#include <src/main/decaf/nio/LongBuffer.h>
Inheritance diagram for decaf::nio::LongBuffer:
```

Public Member Functions

- virtual **~LongBuffer** ()
- virtual std::string **toString** () const
- virtual long long * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the long long array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **LongBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only long long buffer that shares this buffer's content.
- virtual **LongBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **LongBuffer** * **duplicate** ()=0
Creates a new long long buffer that shares this buffer's content.
- virtual long long **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual long long **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **LongBuffer** & **get** (std::vector< long long > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **LongBuffer** & **get** (long long *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible long long array.
- **LongBuffer** & **put** (**LongBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the long longs remaining in the given source buffer long longo this buffer.

- **LongBuffer** & **put** (const long long *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers long longs long longo this buffer from the given source array.

- **LongBuffer** & **put** (std::vector< long long > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source long longs array long longo this buffer.

- virtual **LongBuffer** & **put** (long long value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

Writes the given long longs long longo this buffer at the current position, and then increments the position.

- virtual **LongBuffer** & **put** (std::size_t index, long long value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes the given long longs long longo this buffer at the given index.

- virtual **LongBuffer** * **slice** () const =0

*Creates a new **LongBuffer** (p. 2079) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **LongBuffer** &value) const

Compares this object with the specified object for order.

- virtual bool **equals** (const **LongBuffer** &value) const

- virtual bool **operator==** (const **LongBuffer** &value) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const **LongBuffer** &value) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **LongBuffer** * **allocate** (std::size_t capacity)

Allocates a new Double buffer.

- static **LongBuffer** * **wrap** (long long *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

*Wraps the passed buffer with a new **LongBuffer** (p. 2079).*

- static **LongBuffer** * **wrap** (std::vector< long long > &buffer)

*Wraps the passed STL long long Vector in a **LongBuffer** (p. 2079).*

Protected Member Functions

- **LongBuffer** (std::size_t capacity)

*Creates a **LongBuffer** (p. 2079) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

6.420.1 Detailed Description

This class defines four categories of operations upon long long buffers:.

- o Absolute and relative get and put methods that read and write single long longs;
- o Relative bulk get methods that transfer contiguous sequences of long longs from this buffer long longo an array;
- and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long longo this buffer

o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long longo a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.420.2 Constructor & Destructor Documentation

6.420.2.1 decaf::nio::LongBuffer::LongBuffer (std::size_t capacity) [protected]

Creates a **LongBuffer** (p. 2079) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 803) in doubles

6.420.2.2 virtual decaf::nio::LongBuffer::~~LongBuffer () [inline, virtual]

6.420.3 Member Function Documentation

6.420.3.1 static LongBuffer* decaf::nio::LongBuffer::allocate (std::size_t capacity) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in long longs

Returns:

the **LongBuffer** (p. 2079) that was allocated, caller owns.

6.420.3.2 `virtual long long* decaf::nio::LongBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the long long array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 803)

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2073).

6.420.3.3 `virtual std::size_t decaf::nio::LongBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset long longo the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2074).

6.420.3.4 `virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only long long buffer which the caller then owns.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2074).

6.420.3.5 virtual LongBuffer& decaf::nio::LongBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **LongBuffer** (p. 2079)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2074).

6.420.3.6 virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & value) const [virtual]

Compares this object with the specified object for order. Returns a negative long longeger, zero, or a positive long longeger as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative long longeger, zero, or a positive long longeger as this object is less than, equal to, or greater than the specified object.

6.420.3.7 virtual LongBuffer* decaf::nio::LongBuffer::duplicate () [pure virtual]

Creates a new long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new long long **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2075).

6.420.3.8 `virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & value) const [virtual]`

Returns:

true if this value is considered equal to the passed value.

6.420.3.9 `LongBuffer& decaf::nio::LongBuffer::get (long long * buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)`

Relative bulk get method. This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies `length` long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given `offset` in the array. The position of this buffer is then incremented by `length`.

Parameters:

buffer - long longer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` long longs remaining in this buffer

NullPolong longerException if the passed buffer is null.

6.420.3.10 `LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > buffer) throw (BufferUnderflowException)`

Relative bulk get method. This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than `length` long longs remaining in this buffer

6.420.3.11 `virtual long long decaf::nio::LongBuffer::get (std::size_t index) const
throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 803) where the long long is to be read

Returns:

the long long that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2075).

6.420.3.12 `virtual long long decaf::nio::LongBuffer::get () throw (
BufferUnderflowException) [pure virtual]`

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the long long at the current position

Exceptions:

BufferUnderflowException (p. 837) if there no more data to return

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2076).

6.420.3.13 `virtual bool decaf::nio::LongBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible long long array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2076).

6.420.3.14 `virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer &
value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.420.3.15 `virtual bool decaf::nio::LongBuffer::operator==(const LongBuffer &value) const` [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.420.3.16 `virtual LongBuffer& decaf::nio::LongBuffer::put (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given long longs long longo this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the long longs to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2076).

6.420.3.17 `virtual LongBuffer& decaf::nio::LongBuffer::put (long long value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters:

value - the long longs value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2077).

6.420.3.18 LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source long longs array long longo this buffer. This is the same as calling put(&buffer[0], 0, buffer.size())

Parameters:

buffer - The buffer whose contents are copied to this **LongBuffer** (p. 2079)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.420.3.19 LongBuffer& decaf::nio::LongBuffer::put (const long long * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers long longs long longo this buffer from the given source array. If there are more long longs to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 808), then no long longs are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies length bytes from the given array long longo this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which long longs are to be read

offset- The offset within the array of the first long long to be read;

length - The number of long longs to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.420.3.20 `LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & src)
throw (BufferOverflowException, ReadOnlyBufferException,
lang::exceptions::IllegalArgumentException)`

This method transfers the long longs remaining in the given source buffer long longo this buffer. If there are more long longs remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 808), then no long longs are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters:

src - the buffer to take long longs from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining long longs in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.420.3.21 `virtual LongBuffer* decaf::nio::LongBuffer::slice () const [pure virtual]`

Creates a new **LongBuffer** (p. 2079) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **LongBuffer** (p. 2079) which the caller owns.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 2077).

6.420.3.22 virtual std::string decaf::nio::LongBuffer::toString () const [virtual]**Returns:**

a std::string describing this object

6.420.3.23 static LongBuffer* decaf::nio::LongBuffer::wrap (std::vector< long long > & *buffer*) [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 2079). The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **LongBuffer** (p. 2079) that is backed by buffer, caller owns.

6.420.3.24 static LongBuffer* decaf::nio::LongBuffer::wrap (long long * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **LongBuffer** (p. 2079). The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **LongBuffer** (p. 2079) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

6.421 activemq::util::LongSequenceGenerator Class Reference

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

Public Member Functions

- **LongSequenceGenerator** ()
- virtual **~LongSequenceGenerator** ()
- long long **getNextSequenceId** ()
- long long **getLastSequenceId** ()

6.421.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested. This class is thread safe so the ids can be requested in different threads safely.

6.421.2 Constructor & Destructor Documentation

6.421.2.1 **activemq::util::LongSequenceGenerator::LongSequenceGenerator** ()

6.421.2.2 **virtual**
activemq::util::LongSequenceGenerator::~~LongSequenceGenerator ()
[inline, virtual]

6.421.3 Member Function Documentation

6.421.3.1 **long long activemq::util::LongSequenceGenerator::getLastSequenceId** ()

Returns:

the last id that was generated.

6.421.3.2 **long long activemq::util::LongSequenceGenerator::getNextSequenceId** ()

Returns:

the next id in the sequence.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/LongSequenceGenerator.h`

6.422 decaf::net::MalformedURLException Class Reference

#include <src/main/decaf/net/MalformedURLException.h> Inheritance diagram for decaf::net::MalformedURLException:

Public Member Functions

- **MalformedURLException** () throw ()
Default Constructor.
- **MalformedURLException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **MalformedURLException** (const **MalformedURLException** &ex) throw ()
Copy Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **MalformedURLException** (const std::exception *cause) throw ()
Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **MalformedURLException** * clone () const
Clones this exception.
- virtual ~**MalformedURLException** () throw ()

6.422.1 Constructor & Destructor Documentation

6.422.1.1 decaf::net::MalformedURLException::MalformedURLException () throw () [inline]

Default Constructor.

6.422.1.2 decaf::net::MalformedURLException::MalformedURLException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.422.1.3 `decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.422.1.4 `decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.422.1.5 `decaf::net::MalformedURLException::MalformedURLException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.422.1.6 `decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.422.1.7 virtual decaf::net::MalformedURLException::~~MalformedURLException
() throw () [inline, virtual]

6.422.2 Member Function Documentation

6.422.2.1 virtual MalformedURLException* de-
caf::net::MalformedURLException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**MalformedURLException.h**

6.423 decaf::util::Map< K, V, COMPARATOR > Class Template Reference

Map (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

#include <src/main/decaf/util/Map.h> Inheritance diagram for decaf::util::Map< K, V, COMPARATOR >:

Data Structures

- class **Entry**

Public Member Functions

- **Map** ()
Default constructor - does nothing.
- virtual ~**Map** ()
- virtual bool **equals** (const **Map** &source) const =0
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **Map** &source)=0
Copies the content of the source map into this map.
- virtual void **clear** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
- virtual bool **containsKey** (const K &key) const =0
Indicates whether or this map contains a value for the given key.
- virtual bool **containsValue** (const V &value) const =0
Indicates whether or this map contains a value for the given value, i.e.
- virtual bool **isEmpty** () const =0
- virtual std::size_t **size** () const =0
- virtual V & **get** (const K &key)=0 throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 2094).*
- virtual const V & **get** (const K &key) const =0 throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 2094).*
- virtual void **put** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Sets the value for the specified key.

- virtual void **putAll** (const Map< K, V, COMPARATOR > &other)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Stores a copy of the Mappings contained in the other Map (p. 2094) in this one.
- virtual V **remove** (const K &key)=0 throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)
Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.
- virtual std::vector< K > **keySet** () const =0
Returns a Set (p. 2905) view of the mappings contained in this map.
- virtual std::vector< V > **values** () const =0

6.423.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::Map< K, V, COMPARATOR >

Map (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

6.423.2 Constructor & Destructor Documentation

6.423.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map ()
[inline]

Default constructor - does nothing.

6.423.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~~Map () [inline, virtual]

6.423.3 Member Function Documentation

6.423.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::clear () throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Removes all keys and values from this map.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1106), decaf::util::StlMap< K, V, COMPARATOR > (p. 3034), decaf::util::concurrent::ConcurrentStlMap< Pointer<

MessageId >, Pointer< Message >, MessageId::COMPARATOR >
 (p. 1106), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
 > (p. 1106), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
 > (p. 1106), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1106),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1106),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
 ProducerState >, ProducerId::COMPARATOR > (p. 1106), decaf::util::StlMap<
 cms::Session *, SessionResolver * > (p. 3034), decaf::util::StlMap< std::string,
 WireFormatFactory * > (p. 3034), decaf::util::StlMap< std::string, PrimitiveVal-
 ueNode > (p. 3034), decaf::util::StlMap< std::string, cms::Queue * > (p. 3034),
 decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-
 mands::ProducerId::COMPARATOR > (p. 3034), decaf::util::StlMap< std::string,
 CachedConsumer * > (p. 3034), decaf::util::StlMap< Pointer< commands::ConsumerId
 >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3034),
 decaf::util::StlMap< std::string, TransportFactory * > (p. 3034), decaf::util::StlMap<
 int, Pointer< Command > > (p. 3034), decaf::util::StlMap< Pointer< com-
 mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR
 > (p. 3034), decaf::util::StlMap< std::string, CachedProducer * > (p. 3034), and
 decaf::util::StlMap< std::string, cms::Topic * > (p. 3034).

6.423.3.2 template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR
 >::containsKey (const K & *key*) const [pure virtual]

Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COM-
 PARATOR > (p. 1107), decaf::util::StlMap< K, V, COMPARA-
 TOR > (p. 3034), decaf::util::concurrent::ConcurrentStlMap< Pointer<
 MessageId >, Pointer< Message >, MessageId::COMPARATOR >
 (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
 > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
 > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1107),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1107),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
 ProducerState >, ProducerId::COMPARATOR > (p. 1107), decaf::util::StlMap<
 cms::Session *, SessionResolver * > (p. 3034), decaf::util::StlMap< std::string,
 WireFormatFactory * > (p. 3034), decaf::util::StlMap< std::string, PrimitiveVal-
 ueNode > (p. 3034), decaf::util::StlMap< std::string, cms::Queue * > (p. 3034),

decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3034), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3034), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3034), decaf::util::StlMap< std::string, TransportFactory * > (p. 3034), decaf::util::StlMap< int, Pointer< Command > > (p. 3034), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3034), decaf::util::StlMap< std::string, CachedProducer * > (p. 3034), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3034).

6.423.3.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue (const V & *value*) const [pure virtual]

Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1107), decaf::util::StlMap< K, V, COMPARATOR > (p. 3035), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1107), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1107), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3035), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3035), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3035), decaf::util::StlMap< std::string, cms::Queue * > (p. 3035), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3035), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3035), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3035), decaf::util::StlMap< std::string, TransportFactory * > (p. 3035), decaf::util::StlMap< int, Pointer< Command > > (p. 3035), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3035), decaf::util::StlMap< std::string, CachedProducer * > (p. 3035), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3035).

6.423.3.4 `template<typename K, typename V, typename COMPARTOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARTOR >::copy (const Map< K, V, COMPARTOR > & source) [pure virtual]`

Copies the content of the source map into this map. Erases all existing data in this map.

Parameters:

source The source object to copy from.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR >` (p. 1107), and `decaf::util::StlMap< K, V, COMPARTOR >` (p. 3035).

6.423.3.5 `template<typename K, typename V, typename COMPARTOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARTOR >::equals (const Map< K, V, COMPARTOR > & source) const [pure virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters:

source - `Map` (p. 2094) to compare to this one.

Returns:

true if the `Map` (p. 2094) passed is equal in value to this one.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR >` (p. 1108), and `decaf::util::StlMap< K, V, COMPARTOR >` (p. 3035).

6.423.3.6 `template<typename K, typename V, typename COMPARTOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARTOR >::get (const K & key) const throw (lang::exceptions::NoSuchElementException) [pure virtual]`

Gets the value mapped to the specified key in the `Map` (p. 2094). If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the `Map` (p. 2094).

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR >` (p. 1108), `decaf::util::StlMap< K, V, COMPARTOR >` (p. 3036), `decaf::util::concurrent::ConcurrentStlMap< Pointer<`

```

MessageId >, Pointer< Message >, MessageId::COMPARATOR >
(p. 1108), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
> (p. 1108), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
> (p. 1108), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
>, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1108),
decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,
Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1108),
decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
ProducerState >, ProducerId::COMPARATOR > (p. 1108), decaf::util::StlMap<
cms::Session *, SessionResolver * > (p. 3036), decaf::util::StlMap< std::string,
WireFormatFactory * > (p. 3036), decaf::util::StlMap< std::string, PrimitiveVal-
ueNode > (p. 3036), decaf::util::StlMap< std::string, cms::Queue * > (p. 3036),
decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-
mands::ProducerId::COMPARATOR > (p. 3036), decaf::util::StlMap< std::string,
CachedConsumer * > (p. 3036), decaf::util::StlMap< Pointer< commands::ConsumerId
>, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3036),
decaf::util::StlMap< std::string, TransportFactory * > (p. 3036), decaf::util::StlMap<
int, Pointer< Command > > (p. 3036), decaf::util::StlMap< Pointer< com-
mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR
> (p. 3036), decaf::util::StlMap< std::string, CachedProducer * > (p. 3036), and
decaf::util::StlMap< std::string, cms::Topic * > (p. 3036).

```

6.423.3.7 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual V& decaf::util::Map< K, V, COMPARATOR >::get
(const K & key) throw (lang::exceptions::NoSuchElementException)
[pure virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2094). If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2094).

```

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COM-
PARATOR > (p. 1109), decaf::util::StlMap< K, V, COMPARA-
TOR > (p. 3036), decaf::util::concurrent::ConcurrentStlMap< Pointer<
MessageId >, Pointer< Message >, MessageId::COMPARATOR >
(p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
> (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
> (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
>, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1109),

```

decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1109), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3036), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3036), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3036), decaf::util::StlMap< std::string, cms::Queue * > (p. 3036), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3036), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3036), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3036), decaf::util::StlMap< std::string, TransportFactory * > (p. 3036), decaf::util::StlMap< int, Pointer< Command > > (p. 3036), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3036), decaf::util::StlMap< std::string, CachedProducer * > (p. 3036), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3036).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals().

6.423.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty () const [pure virtual]`

Returns:

if the **Map** (p. 2094) contains any element or not, TRUE or FALSE

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1109), decaf::util::StlMap< K, V, COMPARATOR > (p. 3037), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1109), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1109), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3037), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3037), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3037), decaf::util::StlMap< std::string, cms::Queue * > (p. 3037), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3037), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3037), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3037), decaf::util::StlMap< std::string, TransportFactory * > (p. 3037), decaf::util::StlMap< int, Pointer< Command > > (p. 3037), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3037), decaf::util::StlMap< std::string, CachedProducer * > (p. 3037), and

decaf::util::StlMap< std::string, cms::Topic * > (p. 3037).

6.423.3.9 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual std::vector<K> decaf::util::Map< K, V,
COMPARATOR >::keySet () const [pure virtual]`

Returns a **Set** (p. 2905) view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the set Value operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1833), **Set.remove** (p. 155), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns:

the entire set of keys in this map as a std::vector.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1109), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1109), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1109), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1109), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1109), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1109), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1109), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3037), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3037), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3037), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3037), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3037), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3037), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3037), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3037), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3037), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3037), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3037), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3037).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`.

6.423.3.10 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::put (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sets the value for the specified key.

Parameters:

key The target key.

value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1111), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3038), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1111), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1111), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1111), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1111), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1111), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1111), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3038), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3038), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3038), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3038), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3038), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3038), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3038), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3038), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3038), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3038), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3038), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3038).

6.423.3.11 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2094) in this one.

Parameters:

other A **Map** (p. 2094) instance whose elements are to all be inserted in this **Map** (p. 2094).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1111), decaf::util::StlMap< K, V, COMPARATOR > (p. 3039), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1111), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1111), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1111), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1111), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1111), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1111), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3039), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3039), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3039), decaf::util::StlMap< std::string, cms::Queue * > (p. 3039), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3039), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3039), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3039), decaf::util::StlMap< std::string, TransportFactory * > (p. 3039), decaf::util::StlMap< int, Pointer< Command > > (p. 3039), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3039), decaf::util::StlMap< std::string, CachedProducer * > (p. 3039), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3039).

```
6.423.3.12  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual V decaf::util::Map< K, V,
            COMPARATOR >::remove (const K & key) throw
            ( decaf::lang::exceptions::NoSuchElementException,
            decaf::lang::exceptions::UnsupportedOperationException ) [pure
            virtual]
```

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 2094).

UnsupportedOperationException if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1113), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3039), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1113), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1113), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3039), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3039), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3039), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3039), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3039), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3039), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3039), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3039), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3039), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3039), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3039), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3039).

6.423.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::Map< K, V, COMPARATOR >::size () const [pure virtual]`

Returns:

The number of elements (key/value pairs) in this map.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1114), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3040), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1114), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1114), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3040), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3040), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3040), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3040), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3040), `decaf::util::StlMap< std::string,`

CachedConsumer * > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, TransportFactory * > (p. 3040), decaf::util::StlMap< int, Pointer< Command > > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedProducer * > (p. 3040), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3040).

6.423.3.14 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual std::vector<V> decaf::util::Map< K, V,
COMPARATOR >::values () const [pure virtual]`

Returns:

the entire set of values in this map as a std::vector.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1115), decaf::util::StlMap< K, V, COMPARATOR > (p. 3040), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1115), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3040), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3040), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3040), decaf::util::StlMap< std::string, cms::Queue * > (p. 3040), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, TransportFactory * > (p. 3040), decaf::util::StlMap< int, Pointer< Command > > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedProducer * > (p. 3040), and decaf::util::StlMap< std::string, cms::Topic * > (p. 3040).

The documentation for this class was generated from the following file:

- src/main/decaf/util/Map.h

6.424 cms::MapMessage Class Reference

A **MapMessage** (p. 2106) object is used to send a set of name-value pairs.

#include <src/main/cms/MapMessage.h> Inheritance diagram for cms::MapMessage:

Public Member Functions

- virtual **~MapMessage** ()
- virtual std::vector< std::string > **getMapNames** () const =0 throw (CMSEException)
*Returns an Enumeration of all the names in the **MapMessage** (p. 2106) object.*
- virtual bool **itemExists** (const std::string &name) const =0 throw (CMSEException)
*Indicates whether an item exists in this **MapMessage** (p. 2106) object.*
- virtual bool **getBoolean** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (const std::string &name, bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a boolean value with the specified name into the Map.
- virtual unsigned char **getByte** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Byte value of the Specified name.
- virtual void **setByte** (const std::string &name, unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Byte value with the specified name into the Map.
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Bytes value of the Specified name.
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Bytes value with the specified name into the Map.
- virtual char **getChar** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Char value of the Specified name.
- virtual void **setChar** (const std::string &name, char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Double value of the Specified name.

- virtual void **setDouble** (const std::string &name, double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Double value with the specified name into the Map.

- virtual float **getFloat** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Float value of the Specified name.

- virtual void **setFloat** (const std::string &name, float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Float value with the specified name into the Map.

- virtual int **getInt** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Int value of the Specified name.

- virtual void **setInt** (const std::string &name, int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Int value with the specified name into the Map.

- virtual long long **getLong** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Long value of the Specified name.

- virtual void **setLong** (const std::string &name, long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Long value with the specified name into the Map.

- virtual short **getShort** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Short value of the Specified name.

- virtual void **setShort** (const std::string &name, short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Short value with the specified name into the Map.

- virtual std::string **getString** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the String value of the Specified name.

- virtual void **setString** (const std::string &name, const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a String value with the specified name into the Map.

6.424.1 Detailed Description

A **MapMessage** (p.2106) object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The

names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p. 2106) inherits from the **Message** (p. 2163) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 2106), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 2331) is thrown. To place the **MapMessage** (p. 2106) back into a state where it can be read from and written to, call the `clearBody` method.

MapMessage (p. 2106) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1031). The String-to-primitive conversions may throw a **MessageFormatException** (p. 2278) if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X		X
String	X	X	X		X	X	X	X	X	
byte[]										X

Since:

1.0

6.424.2 Constructor & Destructor Documentation

6.424.2.1 `virtual cms::MapMessage::~~MapMessage () [inline, virtual]`

6.424.3 Member Function Documentation

6.424.3.1 `virtual bool cms::MapMessage::getBoolean (const std::string & name)
const throw (cms::MessageFormatException, cms::CMSEException)
[pure virtual]`

Returns the Boolean value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 312).

6.424.3.2 `virtual unsigned char cms::MapMessage::getByte (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]`

Returns the Byte value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 313).

6.424.3.3 `virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]`

Returns the Bytes value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 313).

6.424.3.4 `virtual char cms::MapMessage::getChar (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]`

Returns the Char value of the Specified name.

Parameters:

name name of the value to fetch from the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 313).

6.424.3.5 virtual double cms::MapMessage::getDouble (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the Double value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 314).

6.424.3.6 virtual float cms::MapMessage::getFloat (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the Float value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 314).

6.424.3.7 virtual int cms::MapMessage::getInt (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the Int value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 314).

6.424.3.8 `virtual long long cms::MapMessage::getLong (const std::string & name)
const throw (cms::MessageFormatException, cms::CMSException)
[pure virtual]`

Returns the Long value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 315).

6.424.3.9 `virtual std::vector< std::string > cms::MapMessage::getMapNames ()
const throw (CMSException) [pure virtual]`

Returns an Enumeration of all the names in the `MapMessage` (p. 2106) object.

Returns:

STL Vector of String values, each of which is the name of an item in the `MapMessage` (p. 2106)

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 315).

6.424.3.10 `virtual short cms::MapMessage::getShort (const std::string & name)
const throw (cms::MessageFormatException, cms::CMSException)
[pure virtual]`

Returns the Short value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 316).

6.424.3.11 `virtual std::string cms::MapMessage::getString (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]`

Returns the String value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 316).

6.424.3.12 `virtual bool cms::MapMessage::itemExists (const std::string & name) const throw (CMSException) [pure virtual]`

Indicates whether an item exists in this `MapMessage` (p. 2106) object.

Parameters:

name String name of the Object in question

Returns:

boolean value indicating if the name is in the map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 316).

6.424.3.13 `virtual void cms::MapMessage::setBoolean (const std::string & name, bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a boolean value with the specified name into the Map.

Parameters:

name the name of the boolean

value the boolean value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageNotWritableException - if the `Message` (p. 2163) is in Read-only Mode.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 317).

6.424.3.14 `virtual void cms::MapMessage::setByte (const std::string & name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException)` [pure virtual]

Sets a Byte value with the specified name into the Map.

Parameters:

name the name of the Byte
value the Byte value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.
MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 317).

6.424.3.15 `virtual void cms::MapMessage::setBytes (const std::string & name, const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException)` [pure virtual]

Sets a Bytes value with the specified name into the Map.

Parameters:

name The name of the Bytes
value The Bytes value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.
MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 318).

6.424.3.16 `virtual void cms::MapMessage::setChar (const std::string & name, char value) throw (cms::MessageNotWriteableException, cms::CMSException)` [pure virtual]

Sets a Char value with the specified name into the Map.

Parameters:

name the name of the Char
value the Char value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 318).

6.424.3.17 virtual void cms::MapMessage::setDouble (const std::string & *name*, double *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Sets a Double value with the specified name into the Map.

Parameters:

name The name of the Double

value The Double value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 318).

6.424.3.18 virtual void cms::MapMessage::setFloat (const std::string & *name*, float *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Sets a Float value with the specified name into the Map.

Parameters:

name The name of the Float

value The Float value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 319).

6.424.3.19 virtual void cms::MapMessage::setInt (const std::string & *name*, int *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Sets a Int value with the specified name into the Map.

Parameters:

name The name of the Int

value The Int value to set in the Map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 319).

6.424.3.20 `virtual void cms::MapMessage::setLong (const std::string & name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)` [pure virtual]

Sets a Long value with the specified name into the Map.

Parameters:

name The name of the Long

value The Long value to set in the Map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 319).

6.424.3.21 `virtual void cms::MapMessage::setShort (const std::string & name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException)` [pure virtual]

Sets a Short value with the specified name into the Map.

Parameters:

name The name of the Short

value The Short value to set in the Map

Exceptions:

CMSEException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 320).

6.424.3.22 `virtual void cms::MapMessage::setString (const std::string & name,
const std::string & value) throw (cms::MessageNotWriteableException,
cms::CMSException)` [pure virtual]

Sets a String value with the specified name into the Map.

Parameters:

name The name of the String

value The String value to set in the Map

Exceptions:

CMSException (p. 1031) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2331) - if the **Message** (p. 2163) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 320).

The documentation for this class was generated from the following file:

- `src/main/cms/MapMessage.h`

6.425 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

Public Member Functions

- **MarkBlockLogger** (**Logger** *logger, const std::string &blockName)
Constructor - Marks Block entry.
- virtual ~**MarkBlockLogger** ()

6.425.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks. Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

6.425.2 Constructor & Destructor Documentation

6.425.2.1 decaf::util::logging::MarkBlockLogger::MarkBlockLogger (**Logger** *logger, const std::string & blockName) [inline]

Constructor - Marks Block entry.

Parameters:

logger **Logger** (p.2028) to use
blockName Block name

6.425.2.2 virtual decaf::util::logging::MarkBlockLogger::~~MarkBlockLogger () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/MarkBlockLogger.h

6.426 activemq::wireformat::MarshalAware Class Reference

#include <src/main/activemq/wireformat/MarshalAware.h> Inheritance diagram for activemq::wireformat::MarshalAware:

Public Member Functions

- virtual `~MarshalAware ()`
- virtual `bool isMarshalAware () const =0`
Determine if the class implementing this interface is really wanting to be told about marshaling.
- virtual `void beforeMarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called before marshaling is started to prepare the object to be marshaled.
- virtual `void afterMarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called after marshaling is started to cleanup the object being marshaled.
- virtual `void beforeUnmarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called before unmarshaling is started to prepare the object to be unmarshaled.
- virtual `void afterUnmarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual `void setMarshaledForm (WireFormat *wireFormat, const std::vector< char > &data)=0`
Called to set the data to this object that will contain the objects marshaled form.
- virtual `std::vector< unsigned char > getMarshaledForm (WireFormat *wireFormat)=0`
Called to get the data to this object that will contain the objects marshaled form.

6.426.1 Constructor & Destructor Documentation

- 6.426.1.1** `virtual activemq::wireformat::MarshalAware::~~MarshalAware ()`
[inline, virtual]

6.426.2 Member Function Documentation

- 6.426.2.1** `virtual void activemq::wireformat::MarshalAware::afterMarshal (WireFormat * wireFormat) throw (decaf::io::IOException)` [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

Parameters:

wireFormat - the wireformat object to control marshaling

6.426.2.2 virtual void activemq::wireformat::MarshalAware::afterUnmarshal
(WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure
virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters:

wireFormat - the wireformat object to control unmarshaling

6.426.2.3 virtual void activemq::wireformat::MarshalAware::beforeMarshal
(WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure
virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters:

wireFormat - the wireformat object to control marshaling

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 311), and `activemq::commands::ActiveMQTextMessage` (p. 574).

6.426.2.4 virtual void activemq::wireformat::MarshalAware::beforeUnmarshal
(WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure
virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

Parameters:

wireFormat - the wireformat object to control unmarshaling

6.426.2.5 virtual std::vector<unsigned char> ac-
tivemq::wireformat::MarshalAware::getMarshaledForm
(WireFormat * *wireFormat*) [pure virtual]

Called to get the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the wireformat object to control unmarshaling

Returns:

buffer that holds the objects data.

6.426.2.6 `virtual bool activemq::wireformat::MarshalAware::isMarshalAware ()` `const [pure virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling. Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns:

true if this class cares about marshaling.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 316), `activemq::commands::BaseDataStructure` (p. 717), `activemq::commands::Message` (p. 2155), and `activemq::commands::WireFormatInfo` (p. 3374).

6.426.2.7 `virtual void activemq::wireformat::MarshalAware::setMarshaledForm` `(WireFormat * wireFormat, const std::vector< char > & data) [pure virtual]`

Called to set the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the wireformat object to control unmarshaling
data - vector of object binary data

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/MarshalAware.h`

6.427 activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.427.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.427.2 Constructor & Destructor Documentation

- 6.427.2.1** virtual
`activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

6.427.3 Member Function Documentation

- 6.427.3.1** virtual void `activemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h`

6.428 activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.428.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.428.2 Constructor & Destructor Documentation

- 6.428.2.1** virtual
`activemq::wireformat::openwire::marshal::v4::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

6.428.3 Member Function Documentation

- 6.428.3.1** virtual void `activemq::wireformat::openwire::marshal::v4::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h`

6.429 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure (OpenWireFormat *format)`

6.429.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.429.2 Constructor & Destructor Documentation

- 6.429.2.1 virtual
activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~MarshallerFactory
() [inline, virtual]

6.429.3 Member Function Documentation

- 6.429.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MarshallerFactory.h**

6.430 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.430.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.430.2 Constructor & Destructor Documentation

- 6.430.2.1** virtual
`activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~MarshallerFactory()` [inline, virtual]

6.430.3 Member Function Documentation

- 6.430.3.1** virtual void `activemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h`

6.431 activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure (OpenWireFormat *format)`

6.431.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.431.2 Constructor & Destructor Documentation

- 6.431.2.1 virtual
activemq::wireformat::openwire::marshal::v5::MarshallerFactory::~MarshallerFactory
() [inline, virtual]

6.431.3 Member Function Documentation

- 6.431.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MarshallerFactory.h**

6.432 decaf::lang::Math Class Reference

The class `Math` (p. 2126) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

```
#include <src/main/decaf/lang/Math.h>
```

Public Member Functions

- `Math ()`
- `virtual ~Math ()`

Static Public Member Functions

- static int **abs** (int value)
Returns the absolute value of an int value.
- static long long **abs** (long long value)
Returns the absolute value of an long long value.
- static float **abs** (float value)
Returns the absolute value of a float value.
- static double **abs** (double value)
Returns the absolute value of a double value.
- static double **sqrt** (double value)
Returns the arc cosine of an angle, in the range of 0.0 through pi.
- static double **pow** (double base, double exp)
Returns the value of the first argument raised to the power of the second argument.
- static short **min** (short a, short b)
Returns the double value that is closest in value to the argument and is equal to a mathematical integer.
- static int **min** (int a, int b)
Returns the smaller of two int values.
- static int **min** (unsigned int a, unsigned int b)
Returns the smaller of two unsigned int values.
- static long long **min** (long long a, long long b)
Returns the smaller of two long long values.
- static float **min** (float a, float b)
Returns the smaller of two float values.
- static double **min** (double a, double b)
Returns the smaller of two double values.

- static short **max** (short a, short b)
*Returns the larger of two **short** values.*
- static int **max** (int a, int b)
*Returns the larger of two **int** values.*
- static long long **max** (long long a, long long b)
*Returns the larger of two **long long** values.*
- static float **max** (float a, float b)
Returns the greater of two float values.
- static double **max** (double a, double b)
Returns the greater of two double values.
- static double **ceil** (double value)
Returns the natural logarithm (base e) of a double value.
- static double **floor** (double value)
Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
- static int **round** (float value)
Returns the closest int to the argument.
- static long long **round** (double value)
Returns the closest long long to the argument.
- static double **random** ()
Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.
- static float **signum** (float value)
Returns Euler's number e raised to the power of a double value.
- static double **signum** (double value)
Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.
- static double **toRadians** (double angdeg)
Returns the measure in radians of the supplied degree angle.
- static double **toDegrees** (double angrad)
Returns the measure in degrees of the supplied radian angle.

Static Public Attributes

- static const double **E**
- static const double **PI**

6.432.1 Detailed Description

The class `Math` (p. 2126) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

6.432.2 Constructor & Destructor Documentation

6.432.2.1 `decaf::lang::Math::Math ()` [inline]

6.432.2.2 `virtual decaf::lang::Math::~~Math ()` [inline, virtual]

6.432.3 Member Function Documentation

6.432.3.1 `static double decaf::lang::Math::abs (double value)` [static]

Returns the absolute value of a double value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

- o If the argument is positive zero or negative zero, the result is positive zero.
- o If the argument is infinite, the result is positive infinity.
- o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: `Double::longBitsToDouble` (p. 1543)(`0x7fffffffffULL & Double::doubleToLongBits(value)`)

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.432.3.2 `static float decaf::lang::Math::abs (float value)` [static]

Returns the absolute value of a float value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

- o If the argument is positive zero or negative zero, the result is positive zero.
- o If the argument is infinite, the result is positive infinity.
- o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: `Float::intBitsToFloat` (p. 1652)(`0x7fffffff & Float::floatToIntBits(value)`)

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.432.3.3 `static long long decaf::lang::Math::abs (long long value)` [inline, static]

Returns the absolute value of an long long value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.432.3.4 static int decaf::lang::Math::abs (int *value*) [inline, static]

Returns the absolute value of an int value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.432.3.5 static double decaf::lang::Math::ceil (double *value*) [static]

Returns the natural logarithm (base e) of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

Parameters:

value the value to compute the natural log of.

Returns:

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to 10ⁿ for integer n, then the result is n.

Parameters:

value - the value to operate on

Returns:

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values x, the result of log1p(x) is much closer to the true result of ln(1 + x) than the floating-point evaluation of log(1.0+x).

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to operate on

Returns:

the the value $\ln(x + 1)$, the natural log of $x + 1$ Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of `Math.ceil(x)` is exactly the value of `-Math.floor(-x)`.

Parameters:

value - the value to find the ceiling of

Returns:

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

6.432.3.6 static double decaf::lang::Math::floor (double *value*) [static]

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the value to find the floor of

Returns:

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

6.432.3.7 static double decaf::lang::Math::max (double *a*, double *b*) [static]

Returns the greater of two double values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of *a* and *b*.

6.432.3.8 static float decaf::lang::Math::max (float *a*, float *b*) [static]

Returns the greater of two float values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of *a* and *b*.

6.432.3.9 static long long decaf::lang::Math::max (long long *a*, long long *b*) [inline, static]

Returns the larger of two long long values. That is, the result the argument closer to the value of `Long::MAX_VALUE` (p.2070). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of *a* and *b*.

6.432.3.10 static int decaf::lang::Math::max (int *a*, int *b*) [inline, static]

Returns the larger of two int values. That is, the result the argument closer to the value of `Integer::MAX_VALUE` (p.1775). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of *a* and *b*.

6.432.3.11 static short decaf::lang::Math::max (short *a*, short *b*) [inline, static]

Returns the larger of two `short` values. That is, the result the argument closer to the value of `Short::MAX_VALUE` (p. 2914). If the arguments have the same value, the result is that same value.

Parameters:

- a* - an argument.
- b* - another argument.

Returns:

the larger of *a* and *b*.

6.432.3.12 static double decaf::lang::Math::min (double *a*, double *b*) [static]

Returns the smaller of two double values. That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters:

- a* - an argument.
- b* - another argument.

Returns:

the smaller of *a* and *b*.

6.432.3.13 static float decaf::lang::Math::min (float *a*, float *b*) [static]

Returns the smaller of two float values. That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters:

- a* - an argument.
- b* - another argument.

Returns:

the smaller of *a* and *b*.

6.432.3.14 static long long decaf::lang::Math::min (long long *a*, long long *b*)
[inline, static]

Returns the smaller of two `long long` values. That is, the result the argument closer to the value of `Long::MIN_VALUE` (p.2070). If the arguments have the same value, the result is that same value.

Parameters:

- a* - an argument.
- b* - another argument.

Returns:

the smaller of *a* and *b*.

6.432.3.15 static int decaf::lang::Math::min (unsigned int *a*, unsigned int *b*)
[inline, static]

Returns the smaller of two `unsigned int` values. That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p.1776). If the arguments have the same value, the result is that same value.

Parameters:

- a* - an argument.
- b* - another argument.

Returns:

the smaller of *a* and *b*.

6.432.3.16 static int decaf::lang::Math::min (int *a*, int *b*) [inline, static]

Returns the smaller of two `int` values. That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p.1776). If the arguments have the same value, the result is that same value.

Parameters:

- a* - an argument.
- b* - another argument.

Returns:

the smaller of *a* and *b*.

6.432.3.17 static short decaf::lang::Math::min (short *a*, short *b*) [inline, static]

Returns the double value that is closest in value to the argument and is equal to a mathematical integer. If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the value to round to the nearest integer

Returns:

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of `decaf::lang::Short::MIN_VALUE` (p. 2914). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of *a* and *b*.

6.432.3.18 `static double decaf::lang::Math::pow (double base, double exp)` [static]

Returns the value of the first argument raised to the power of the second argument. Special cases:

o If the second argument is positive or negative zero, then the result is 1.0. o If the second argument is 1.0, then the result is the same as the first argument. o If the second argument is NaN, then the result is NaN. o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

Parameters:

base - the base

exp - the exponent

Returns:

the base raised to the power of *exp*.

6.432.3.19 `static double decaf::lang::Math::random ()` [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard. The remainder value is mathematically equal to $f1 - f2 \times n$, where *n* is the mathematical integer closest to the exact mathematical value of the quotient $f1/f2$, and if two mathematical integers are equally close to $f1/f2$, then *n* is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN. o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

Parameters:*f1* - the dividend.*f2* - the divisor**Returns:**

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

Returns:

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

6.432.3.20 static long long decaf::lang::Math::round (double *value*) [static]

Returns the closest long long to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long) **Math.floor** (p. 2130)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN_VALUE** (p. 2070), the result is equal to the value of **Long::MIN_VALUE** (p. 2070). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX_VALUE** (p. 2070), the result is equal to the value of **Long::MAX_VALUE** (p. 2070).

Parameters:*value* - the value to round**Returns:**

the value of the argument rounded to the nearest integral value.

6.432.3.21 static int decaf::lang::Math::round (float *value*) [static]

Returns the closest int to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int) **Math.floor** (p. 2130)(a + 0.5f)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN_VALUE** (p. 1776), the result is equal to the value of **Integer::MIN_VALUE** (p. 1776). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX_VALUE** (p. 1775), the result is equal to the value of **Integer::MAX_VALUE** (p. 1775).

Parameters:

value - the value to round

Returns:

the value of the argument rounded to the nearest integral value.

6.432.3.22 static double decaf::lang::Math::signum (double *value*) [static]

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the floating-point value whose signum is to be returned

Returns:

the signum function of the argument

6.432.3.23 static float decaf::lang::Math::signum (float *value*) [static]

Returns Euler's number e raised to the power of a double value. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

Parameters:

value - the exponent to raise e to

Returns:

the value e^a , where e is the base of the natural logarithms. Returns $e^x - 1$. Note that for values of x near 0, the exact sum of $\expm1(x) + 1$ is much closer to the true result of e^x than $\exp(x)$. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to raise $e^x - 1$

Returns:

the value $e^x - 1$. Returns $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

Parameters:

x - an argument

y - another argument

Returns:

the $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0 if the argument is greater than zero, -1.0 if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the floating-point value whose signum is to be returned

Returns:

the signum function of the argument

6.432.3.24 static double decaf::lang::Math::sqrt (double *value*) [static]

Returns the arc cosine of an angle, in the range of 0.0 through pi. Special case:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

Parameters:

value - the value to return the arc cosine of.

Returns:

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to return the arc cosine of.

Returns:

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to return the arc cosine of.

Returns:

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3*pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3*pi/4.

Parameters:

y - the ordinate coordinate
x - the abscissa coordinate

Returns:

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x, `cbrt(-x) == -cbrt(x)`; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the double to compute the cube root of

Returns:

the cube root of value Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

Parameters:

value - an value in radians

Returns:

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of x is defined to be $(e^x + e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

Parameters:

value - the number whose hyperbolic cosine is to be found

Returns:

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the number whose sin is to be found

Returns:

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the number whose hyperbolic sin is to be found

Returns:

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the number whose tangent is to be found

Returns:

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x})/(e^x + e^{-x})$, in other words, $\sinh(x)/\cosh(x)$. Note that the absolute value of the exact tanh is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

Parameters:

value - the number whose hyperbolic tangent is to be found

Returns:

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

Parameters:

value - the value to find the square root of
the square root of the argument.

6.432.3.25 `static double decaf::lang::Math::toDegrees (double angrad)` [inline, static]

Returns the measure in degrees of the supplied radian angle.

Parameters:

angrad - an angle in radians

Returns:

the degree measure of the angle.

6.432.3.26 `static double decaf::lang::Math::toRadians (double angdeg)` [inline, static]

Returns the measure in radians of the supplied degree angle.

Parameters:

angdeg - an angle in degrees

Returns:

the radian measure of the angle.

6.432.4 Field Documentation

6.432.4.1 `const double decaf::lang::Math::E` [static]

6.432.4.2 `const double decaf::lang::Math::PI` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

6.433 activemq::util::MemoryUsage Class Reference

#include <src/main/activemq/util/MemoryUsage.h> Inheritance diagram for activemq::util::MemoryUsage:

Public Member Functions

- **MemoryUsage** ()
Default Constructor.
- **MemoryUsage** (unsigned long long limit)
*Creates an instance of an **Usage** (p. 3351) monitor with a set limit.*
- virtual ~**MemoryUsage** ()
- virtual void **waitForSpace** ()
*Waits forever for more space to be returned to this **Usage** (p. 3351) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)
*Waits for more space to be returned to this **Usage** (p. 3351) Manager, times out when the given time span in milliseconds elapses.*
- virtual void **enqueueUsage** (unsigned long long value)
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void **increaseUsage** (unsigned long long value)
Increases the usage by the value amount.
- virtual void **decreaseUsage** (unsigned long long value)
Decreases the usage by the value amount.
- virtual bool **isFull** () const
*Returns true if this **Usage** (p. 3351) instance is full, i.e.*
- unsigned long long **getUsage** () const
Gets the current usage amount.
- void **setUsage** (unsigned long long usage)
Sets the current usage amount.
- unsigned long long **getLimit** () const
Gets the current limit amount.
- void **setLimit** (unsigned long long limit)
Sets the current limit amount.

6.433.1 Constructor & Destructor Documentation

6.433.1.1 `activemq::util::MemoryUsage::MemoryUsage ()`

Default Constructor.

6.433.1.2 `activemq::util::MemoryUsage::MemoryUsage (unsigned long long limit)`

Creates an instance of an **Usage** (p. 3351) monitor with a set limit.

Parameters:

limit - amount of memory this manager allows.

6.433.1.3 `virtual activemq::util::MemoryUsage::~~MemoryUsage ()` [virtual]

6.433.2 Member Function Documentation

6.433.2.1 `virtual void activemq::util::MemoryUsage::decreaseUsage (unsigned long long value)` [virtual]

Decreases the usage by the value amount.

Parameters:

value Amount of space to return to the pool

Implements **activemq::util::Usage** (p. 3351).

6.433.2.2 `virtual void activemq::util::MemoryUsage::enqueueUsage (unsigned long long value)` [inline, virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters:

value Amount of usage in bytes to add.

Implements **activemq::util::Usage** (p. 3351).

6.433.2.3 `unsigned long long activemq::util::MemoryUsage::getLimit ()` const [inline]

Gets the current limit amount.

Returns:

the amount that can be used before full.

6.433.2.4 `unsigned long long activemq::util::MemoryUsage::getUsage () const`
 `[inline]`

Gets the current usage amount.

Returns:

the amount of bytes currently used.

6.433.2.5 `virtual void activemq::util::MemoryUsage::increaseUsage (unsigned long`
 `long value) [virtual]`

Increases the usage by the value amount.

Parameters:

value Amount of usage to add.

Implements `activemq::util::Usage` (p. 3352).

6.433.2.6 `virtual bool activemq::util::MemoryUsage::isFull () const [virtual]`

Returns true if this `Usage` (p. 3351) instance is full, i.e. `Usage` (p. 3351) $\geq 100\%$

Implements `activemq::util::Usage` (p. 3352).

6.433.2.7 `void activemq::util::MemoryUsage::setLimit (unsigned long long limit)`
 `[inline]`

Sets the current limit amount.

Parameters:

limit - The amount that can be used before full.

6.433.2.8 `void activemq::util::MemoryUsage::setUsage (unsigned long long usage)`
 `[inline]`

Sets the current usage amount.

Parameters:

usage - The amount to tag as used.

6.433.2.9 `virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int`
 `timeout) [virtual]`

Waits for more space to be returned to this `Usage` (p. 3351) Manager, times out when the given time span in milliseconds elapses.

Parameters:

timeout The time to wait for more space.

Implements `activemq::util::Usage` (p. 3352).

6.433.2.10 `virtual void activemq::util::MemoryUsage::waitForSpace ()` [virtual]

Waits forever for more space to be returned to this **Usage** (p. 3351) Manager.

Implements **activemq::util::Usage** (p. 3352).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/MemoryUsage.h`

6.434 activemq::commands::Message Class Reference

#include <src/main/activemq/commands/Message.h> Inheritance diagram for activemq::commands::Message:

Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **Message * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)
Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.
- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)
*Sets the Acknowledgment Handler that this **Message** (p. 2145) will use when the Acknowledge method is called.*
- virtual **Pointer**< **core::ActiveMQAckHandler** > **getAckHandler** () const
*Gets the Acknowledgment Handler that this **Message** (p. 2145) will use when the Acknowledge method is called.*
- virtual unsigned int **getSize** () const

Returns the Size of this message in Bytes.

- virtual bool **isExpired** () const
Returns if this message has expired, meaning that its Expiration time has elapsed.
- virtual void **onSend** ()
*Allows derived **Message** (p. 2145) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap** & **getMessageProperties** ()
Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.
- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const
*Returns if the **Message** (p. 2145) Properties Are Read Only.*
- void **setReadOnlyProperties** (bool value)
*Set the Read Only State of the **Message** (p. 2145) Properties.*
- bool **isReadOnlyBody** () const
*Returns if the **Message** (p. 2145) Body is Read Only.*
- void **setReadOnlyBody** (bool value)
*Set the Read Only State of the **Message** (p. 2145) Content.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** ()
- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &originalDestination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual const **Pointer**< **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()
- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &originalTransactionId)
- virtual const std::string & **getGroupID** () const
- virtual std::string & **getGroupID** ()
- virtual void **setGroupID** (const std::string &groupID)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int groupSequence)

- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &correlationId)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool persistent)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long expiration)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char priority)
- virtual const **Pointer**< **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &replyTo)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long timestamp)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &type)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &content)
- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &marshalled-Properties)
- virtual const **Pointer**< **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &dataStructure)
- virtual const **Pointer**< **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &targetConsumerId)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool compressed)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual const std::vector< decaf::lang::Pointer< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< decaf::lang::Pointer< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< decaf::lang::Pointer< **BrokerId** > > &brokerPath)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long arrival)
- virtual const std::string & **getUserID** () const
- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &userID)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool recievedByDFBridge)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool droppable)
- virtual const std::vector< decaf::lang::Pointer< **BrokerId** > > & **getCluster** () const

- virtual std::vector< **decaf::lang::Pointer< BrokerId >** > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer< BrokerId >** > &cluster)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long brokerInTime)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long brokerOutTime)
- virtual bool **isMessage** () const
- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _MESSAGE** = 0

Protected Member Functions

- **Message** (const **Message** &)
- **Message** & **operator=** (const **Message** &)

Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- **Pointer< TransactionId > transactionId**
- **Pointer< ActiveMQDestination > originalDestination**
- **Pointer< MessageId > messageId**
- **Pointer< TransactionId > originalTransactionId**
- std::string **groupID**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer< ActiveMQDestination > replyTo**
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**
- **Pointer< DataStructure > dataStructure**
- **Pointer< ConsumerId > targetConsumerId**
- bool **compressed**
- int **redeliveryCounter**
- std::vector< **decaf::lang::Pointer< BrokerId >** > **brokerPath**
- long long **arrival**
- std::string **userID**

- bool **recievedByDFBridge**
- bool **droppable**
- std::vector< **decaf::lang::Pointer< BrokerId > > cluster**
- long long **brokerInTime**
- long long **brokerOutTime**

Static Protected Attributes

- static const unsigned int **DEFAULT_MESSAGE_SIZE** = 1024

6.434.1 Constructor & Destructor Documentation

6.434.1.1 `activemq::commands::Message::Message (const Message &) [inline, protected]`

6.434.1.2 `activemq::commands::Message::Message ()`

6.434.1.3 `virtual activemq::commands::Message::~~Message () [virtual]`

6.434.2 Member Function Documentation

6.434.2.1 `virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [virtual]`

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters:

wireFormat - the wireformat object to control unmarshaling

Reimplemented from `activemq::commands::BaseDataStructure` (p. 716).

6.434.2.2 `virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [virtual]`

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

Parameters:

wireFormat - the wireformat controller

Reimplemented from `activemq::commands::BaseDataStructure` (p. 716).

6.434.2.3 `virtual Message* activemq::commands::Message::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 173), `activemq::commands::ActiveMQBytesMessage` (p. 200), `activemq::commands::ActiveMQMapMessage` (p. 312), `activemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQObjectMessage` (p. 384), `activemq::commands::ActiveMQStreamMessage` (p. 467), and `activemq::commands::ActiveMQTextMessage` (p. 575).

6.434.2.4 `virtual void activemq::commands::Message::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 173), `activemq::commands::ActiveMQBytesMessage` (p. 201), `activemq::commands::ActiveMQMapMessage` (p. 312), `activemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQObjectMessage` (p. 384), `activemq::commands::ActiveMQStreamMessage` (p. 467), and `activemq::commands::ActiveMQTextMessage` (p. 575).

6.434.2.5 `virtual bool activemq::commands::Message::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 174), `activemq::commands::ActiveMQBytesMessage` (p. 201), `activemq::commands::ActiveMQMapMessage` (p. 312), `activemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQMessageTemplate< T >` (p. 369), `activemq::commands::ActiveMQObjectMessage` (p. 384), `activemq::commands::ActiveMQStreamMessage` (p. 468), `activemq::commands::ActiveMQTextMessage` (p. 575), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 369), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 369).

```
6.434.2.6 virtual Pointer<core::ActiveMQAckHandler>
activemq::commands::Message::getAckHandler () const [inline,
virtual]
```

Gets the Acknowledgment Handler that this **Message** (p. 2145) will use when the Acknowledge method is called.

Returns:

handler ActiveMQAckHandler to call or NULL if not set

- 6.434.2.7 `virtual long long activemq::commands::Message::getArrival () const [virtual]`
- 6.434.2.8 `virtual long long activemq::commands::Message::getBrokerInTime () const [virtual]`
- 6.434.2.9 `virtual long long activemq::commands::Message::getBrokerOutTime () const [virtual]`
- 6.434.2.10 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () [virtual]`
- 6.434.2.11 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () const [virtual]`
- 6.434.2.12 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () [virtual]`
- 6.434.2.13 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () const [virtual]`
- 6.434.2.14 `virtual std::vector<unsigned char>& activemq::commands::Message::getContent () [virtual]`
- 6.434.2.15 `virtual const std::vector<unsigned char>& activemq::commands::Message::getContent () const [virtual]`
- 6.434.2.16 `virtual std::string& activemq::commands::Message::getCorrelationId () [virtual]`
- 6.434.2.17 `virtual const std::string& activemq::commands::Message::getCorrelationId () const [virtual]`
- 6.434.2.18 `virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure () [virtual]`
- 6.434.2.19 `virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure () const [virtual]`
- 6.434.2.20 `virtual unsigned char activemq::commands::Message::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 174), `activemq::commands::ActiveMQBytesMessage` (p. 202), `activemq::commands::ActiveMQMapMessage` (p. 314), `activemq::commands::ActiveMQMessage` (p. 343), `activemq::commands::ActiveMQObjectMessage` (p. 385), `activemq::commands::ActiveMQStreamMessage` (p. 468), and `activemq::commands::ActiveMQTextMessage` (p. 575).

- 6.434.2.21** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () [virtual]`
- 6.434.2.22** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () const [virtual]`
- 6.434.2.23** `virtual long long activemq::commands::Message::getExpiration () const [virtual]`
- 6.434.2.24** `virtual std::string& activemq::commands::Message::getGroupID () [virtual]`
- 6.434.2.25** `virtual const std::string& activemq::commands::Message::getGroupID () const [virtual]`
- 6.434.2.26** `virtual int activemq::commands::Message::getGroupSequence () const [virtual]`
- 6.434.2.27** `virtual std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () [virtual]`
- 6.434.2.28** `virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () const [virtual]`
- 6.434.2.29** `virtual Pointer<MessageId>& activemq::commands::Message::getMessageId () [virtual]`
- 6.434.2.30** `virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId () const [virtual]`
- 6.434.2.31** `const util::PrimitiveMap& activemq::commands::Message::getMessageProperties () const [inline]`
- 6.434.2.32** `util::PrimitiveMap& activemq::commands::Message::getMessageProperties () [inline]`

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

Returns:

a reference to the Primitive Map that holds message properties.

- 6.434.2.33** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ()`
[virtual]
- 6.434.2.34** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () const`
[virtual]
- 6.434.2.35** `virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ()`
[virtual]
- 6.434.2.36** `virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () const`
[virtual]
- 6.434.2.37** `virtual unsigned char activemq::commands::Message::getPriority ()`
const [virtual]
- 6.434.2.38** `virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ()`
[virtual]
- 6.434.2.39** `virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId () const`
[virtual]
- 6.434.2.40** `virtual int activemq::commands::Message::getRedeliveryCounter ()`
const [virtual]
- 6.434.2.41** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ()` [virtual]
- 6.434.2.42** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () const` [virtual]
- 6.434.2.43** `virtual unsigned int activemq::commands::Message::getSize () const`
[virtual]

Returns the Size of this message in Bytes.

Returns:

number of bytes this message equates to.

Reimplemented in `activemq::commands::ActiveMQTextMessage` (p. 576).

- 6.434.2.44** `virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ()`
[virtual]
- 6.434.2.45** `virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ()`
const [virtual]
- 6.434.2.46** `virtual long long activemq::commands::Message::getTimestamp ()` const
[virtual]
- 6.434.2.47** `virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId ()`
[virtual]
- 6.434.2.48** `virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId ()` const
[virtual]
- 6.434.2.49** `virtual std::string& activemq::commands::Message::getType ()`
[virtual]
- 6.434.2.50** `virtual const std::string& activemq::commands::Message::getType ()`
const [virtual]
- 6.434.2.51** `virtual std::string& activemq::commands::Message::getUserID ()`
[virtual]
- 6.434.2.52** `virtual const std::string& activemq::commands::Message::getUserID ()`
const [virtual]
- 6.434.2.53** `virtual bool activemq::commands::Message::isCompressed ()` const
[virtual]
- 6.434.2.54** `virtual bool activemq::commands::Message::isDroppable ()` const
[virtual]
- 6.434.2.55** `virtual bool activemq::commands::Message::isExpired ()` const
[virtual]

Returns if this message has expired, meaning that its Expiration time has elapsed.

Returns:

true if message is expired.

- 6.434.2.56** `virtual bool activemq::commands::Message::isMarshalAware ()` const
[inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns:

boolean indicating desire to be in marshaling stages

Reimplemented from `activemq::commands::BaseDataStructure` (p. 717).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 316).

6.434.2.57 `virtual bool activemq::commands::Message::isMessage () const`
[inline, virtual]

Returns:

an answer of true to the `isMessage()` (p. 2156) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 653).

6.434.2.58 `virtual bool activemq::commands::Message::isPersistent () const`
[virtual]

6.434.2.59 `bool activemq::commands::Message::isReadOnlyBody () const` [inline]

Returns if the `Message` (p. 2145) Body is Read Only.

Returns:

true if `Message` (p. 2145) Content is Read Only.

6.434.2.60 `bool activemq::commands::Message::isReadOnlyProperties () const`
[inline]

Returns if the `Message` (p. 2145) Properties Are Read Only.

Returns:

true if `Message` (p. 2145) Properties are Read Only.

6.434.2.61 `virtual bool activemq::commands::Message::isRecievedByDFBridge ()`
`const` [virtual]

6.434.2.62 `virtual void activemq::commands::Message::onSend ()` [inline,
virtual]

Allows derived `Message` (p. 2145) classes to perform tasks before a message is sent.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 202),
`activemq::commands::ActiveMQMessageTemplate< T >` (p. 376),
`activemq::commands::ActiveMQStreamMessage` (p. 468),
`activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 376),
`activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 376),
`activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 376),
`activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 376),
`activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 376), and
`activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 376).

6.434.2.63 `Message& activemq::commands::Message::operator= (const Message &)`
[inline, protected]

6.434.2.64 `virtual void activemq::commands::Message::setAckHandler (const
Pointer< core::ActiveMQAckHandler > & handler)` [inline, virtual]

Sets the Acknowledgment Handler that this **Message** (p.2145) will use when the Acknowledge method is called.

Parameters:

handler ActiveMQAckHandler to call

- 6.434.2.65 virtual void activemq::commands::Message::setArrival (long long *arrival*) [virtual]
- 6.434.2.66 virtual void activemq::commands::Message::setBrokerInTime (long long *brokerInTime*) [virtual]
- 6.434.2.67 virtual void activemq::commands::Message::setBrokerOutTime (long long *brokerOutTime*) [virtual]
- 6.434.2.68 virtual void activemq::commands::Message::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]
- 6.434.2.69 virtual void activemq::commands::Message::setCluster (const std::vector< decaf::lang::Pointer< BrokerId > > & *cluster*) [virtual]
- 6.434.2.70 virtual void activemq::commands::Message::setCompressed (bool *compressed*) [virtual]
- 6.434.2.71 virtual void activemq::commands::Message::setContent (const std::vector< unsigned char > & *content*) [virtual]
- 6.434.2.72 virtual void activemq::commands::Message::setCorrelationId (const std::string & *correlationId*) [virtual]
- 6.434.2.73 virtual void activemq::commands::Message::setDataStructure (const Pointer< DataStructure > & *dataStructure*) [virtual]
- 6.434.2.74 virtual void activemq::commands::Message::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.434.2.75 virtual void activemq::commands::Message::setDroppable (bool *droppable*) [virtual]
- 6.434.2.76 virtual void activemq::commands::Message::setExpiration (long long *expiration*) [virtual]
- 6.434.2.77 virtual void activemq::commands::Message::setGroupID (const std::string & *groupID*) [virtual]
- 6.434.2.78 virtual void activemq::commands::Message::setGroupSequence (int *groupSequence*) [virtual]
- 6.434.2.79 virtual void activemq::commands::Message::setMarshaledProperties (const std::vector< unsigned char > & *marshalledProperties*) [virtual]
- 6.434.2.80 virtual void activemq::commands::Message::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.434.2.81 virtual void activemq::commands::Message::setOriginalDestination (const Pointer< ActiveMQDestination > & *originalDestination*) [virtual]
- 6.434.2.82 virtual void activemq::commands::Message::setOriginalTransactionId (const Pointer< TransactionId > & *originalTransactionId*) [virtual]

Generated on Fri Apr 30 14:04:20 2010 for activemq-cpp-3.1.2 by Doxygen

- 6.434.2.83 virtual void activemq::commands::Message::setPersistent (bool *persistent*) [virtual]
- 6.434.2.84 virtual void activemq::commands::Message::setPriority (unsigned char *priority*) [virtual]

Parameters:

value - true if Content should be read only.

6.434.2.87 void activemq::commands::Message::setReadOnlyProperties (bool *value*) [inline]

Set the Read Only State of the **Message** (p. 2145) Properties.

Parameters:

value - true if Properties should be read only.

6.434.2.88 virtual void activemq::commands::Message::setRecievedByDFBridge (bool *recievedByDFBridge*) [virtual]

6.434.2.89 virtual void activemq::commands::Message::setRedeliveryCounter (int *redeliveryCounter*) [virtual]

6.434.2.90 virtual void activemq::commands::Message::setReplyTo (const Pointer< ActiveMQDestination > & *replyTo*) [virtual]

6.434.2.91 virtual void activemq::commands::Message::setTargetConsumerId (const Pointer< ConsumerId > & *targetConsumerId*) [virtual]

6.434.2.92 virtual void activemq::commands::Message::setTimestamp (long long *timestamp*) [virtual]

6.434.2.93 virtual void activemq::commands::Message::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.434.2.94 virtual void activemq::commands::Message::setType (const std::string & *type*) [virtual]

6.434.2.95 virtual void activemq::commands::Message::setUserID (const std::string & *userID*) [virtual]

6.434.2.96 virtual std::string activemq::commands::Message::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 176), **activemq::commands::ActiveMQBytesMessage** (p. 208), **activemq::commands::ActiveMQMapMessage** (p. 320), **activemq::commands::ActiveMQMessage** (p. 344), **activemq::commands::ActiveMQObjectMessage** (p. 385), **activemq::commands::ActiveMQStreamMessage** (p. 474), and **activemq::commands::ActiveMQTextMessage** (p. 577).

6.434.2.97 `virtual Pointer<Command> activemq::commands::Message::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.434.3 Field Documentation

- 6.434.3.1 `long long activemq::commands::Message::arrival` [protected]
- 6.434.3.2 `long long activemq::commands::Message::brokerInTime` [protected]
- 6.434.3.3 `long long activemq::commands::Message::brokerOutTime` [protected]
- 6.434.3.4 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::brokerPath` [protected]
- 6.434.3.5 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::cluster` [protected]
- 6.434.3.6 `bool activemq::commands::Message::compressed` [protected]
- 6.434.3.7 `std::vector<unsigned char> activemq::commands::Message::content` [protected]
- 6.434.3.8 `std::string activemq::commands::Message::correlationId` [protected]
- 6.434.3.9 `Pointer<DataStructure> activemq::commands::Message::dataStructure` [protected]
- 6.434.3.10 `const unsigned int activemq::commands::Message::DEFAULT_MESSAGE_SIZE = 1024` [static, protected]
- 6.434.3.11 `Pointer<ActiveMQDestination> activemq::commands::Message::destination` [protected]
- 6.434.3.12 `bool activemq::commands::Message::droppable` [protected]
- 6.434.3.13 `long long activemq::commands::Message::expiration` [protected]
- 6.434.3.14 `std::string activemq::commands::Message::groupId` [protected]
- 6.434.3.15 `int activemq::commands::Message::groupSequence` [protected]
- 6.434.3.16 `const unsigned char activemq::commands::Message::ID_MESSAGE = 0` [static]
- 6.434.3.17 `std::vector<unsigned char> activemq::commands::Message::marshalledProperties` [protected]
- 6.434.3.18 `Pointer<MessageId> activemq::commands::Message::messageId` [protected]
- 6.434.3.19 `Pointer<ActiveMQDestination> activemq::commands::Message::originalDestination` [protected]
- 6.434.3.20 `Pointer<TransactionId> activemq::commands::Message::originalTransactionId` [protected]

Generated on Fri Apr 30 14:04:20 2010 for activemq-cpp-3.1.2 by Doxygen

- 6.434.3.21 `bool activemq::commands::Message::persistent` [protected]
- 6.434.3.22 `unsigned char activemq::commands::Message::priority` [protected]

- 6.434.3.23 `Pointer<ProducerId> activemq::commands::Message::producerId` [protected]

- `src/main/activemq/commands/Message.h`

6.435 cms::Message Class Reference

Root of all messages.

#include <src/main/cms/Message.h> Inheritance diagram for cms::Message:

Public Member Functions

- virtual `~Message ()`
- virtual `Message * clone () const =0`
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void `acknowledge () const =0 throw (IllegalStateException, CMSEException)`
Acknowledges all consumed messages of the session of this consumed message.
- virtual void `clearBody ()=0 throw (CMSEException)`
Clears out the body of the message.
- virtual void `clearProperties ()=0 throw (CMSEException)`
Clears out the message body.
- virtual `std::vector< std::string > getPropertyNames () const =0 throw (CMSEException)`
Retrieves the property names.
- virtual bool `propertyExists (const std::string &name) const =0 throw (CMSEException)`
Indicates whether or not a given property exists.
- virtual bool `getBooleanProperty (const std::string &name) const =0 throw (MessageFormatException, CMSEException)`
Gets a boolean property.
- virtual unsigned char `getByteProperty (const std::string &name) const =0 throw (MessageFormatException, CMSEException)`
Gets a byte property.
- virtual double `getDoubleProperty (const std::string &name) const =0 throw (MessageFormatException, CMSEException)`
Gets a double property.
- virtual float `getFloatProperty (const std::string &name) const =0 throw (MessageFormatException, CMSEException)`
Gets a float property.
- virtual int `getIntProperty (const std::string &name) const =0 throw (MessageFormatException, CMSEException)`
Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a string property.
- virtual std::string **getCMSCorrelationID** () const =0 throw (CMSEException)
Gets the correlation ID for the message.
- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw (CMSEException)
Sets the correlation ID for the message.

- virtual int **getCMSDeliveryMode** () const =0 throw (CMSEException)
*Gets the **DeliveryMode** (p. 1478) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0 throw (CMSEException)
*Sets the **DeliveryMode** (p. 1478) for this message.*
- virtual const **Destination** * **getCMSDestination** () const =0 throw (CMSEException)
*Gets the **Destination** (p. 1480) object for this message.*
- virtual void **setCMSDestination** (const **Destination** *destination)=0 throw (CMSEException)
*Sets the **Destination** (p. 1480) object for this message.*
- virtual long long **getCMSExpiration** () const =0 throw (CMSEException)
Gets the message's expiration value.
- virtual void **setCMSExpiration** (long long expireTime)=0 throw (CMSEException)
Sets the message's expiration value.
- virtual std::string **getCMSMessageID** () const =0 throw (CMSEException)
The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.
- virtual void **setCMSMessageID** (const std::string &id)=0 throw (CMSEException)
Sets the message ID.
- virtual int **getCMSPriority** () const =0 throw (CMSEException)
Gets the message priority level.
- virtual void **setCMSPriority** (int priority)=0 throw (CMSEException)
Sets the Priority Value for this message.
- virtual bool **getCMSRedelivered** () const =0 throw (CMSEException)
Gets an indication of whether this message is being redelivered.
- virtual void **setCMSRedelivered** (bool redelivered)=0 throw (CMSEException)
Specifies whether this message is being redelivered.
- virtual const **cms::Destination** * **getCMSReplyTo** () const =0 throw (CMSEException)
*Gets the **Destination** (p. 1480) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** *destination)=0 throw (CMSEException)
*Sets the **Destination** (p. 1480) object to which a reply to this message should be sent.*
- virtual long long **getCMSTimestamp** () const =0 throw (CMSEException)
Gets the message timestamp.
- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw (CMSEException)

Sets the message timestamp.

- virtual std::string **getCMSType** () const =0 throw (CMSEException)

Gets the message type identifier supplied by the client when the message was sent.

- virtual void **setCMSType** (const std::string &type)=0 throw (CMSEException)

Sets the message type.

6.435.1 Detailed Description

Root of all messages. As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

Message (p. 2163) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 2163) Interface definition.

- Stream - A **StreamMessage** (p. 3081) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 938) type the values written to a **StreamMessage** (p. 3081) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 2163) Body.
- Map - A **MapMessage** (p. 2106) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 2106) makes no guarantee on the order of the elements within the **Message** (p. 2163) body.
- Text - A **TextMessage** (p. 3170) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 938) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

Message (p. 2163) Properties

Message (p. 2163) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1031). The String-to-primitive conversions may throw a runtime exception if the primitive's valueOf method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X
short			X	X	X			X
int				X	X			X
long					X			X
float						X	X	X
double							X	X
String	X	X	X	X	X	X	X	X

When a **Message** (p. 2163) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered Message's properties will result in a **CMSEException** (p. 1031) being thrown.

See also:

JMS API

Since:

1.0

6.435.2 Constructor & Destructor Documentation

6.435.2.1 virtual cms::Message::~Message () [inline, virtual]

6.435.3 Member Function Documentation

6.435.3.1 virtual void cms::Message::acknowledge () const throw (
 IllegalStateException, **CMSEException**) [pure virtual]

Acknowledges all consumed messages of the session of this consumed message. All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

IllegalStateException (p. 1729) - if this method is called on a closed session.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 368), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 368), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 368), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 368), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 368), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 368).

6.435.3.2 virtual void cms::Message::clearBody () throw (**CMSEException**) [pure virtual]

Clears out the body of the message. This does not clear the headers or properties.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 200), `activemq::commands::ActiveMQMapMessage` (p. 311), `activemq::commands::ActiveMQStreamMessage` (p. 467), `activemq::commands::ActiveMQTextMessage` (p. 574), `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 368), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 368), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 368), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 368), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 368), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 368).

6.435.3.3 `virtual void cms::Message::clearProperties () throw (CMSEException)` [pure virtual]

Clears out the message body. Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 369), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 369), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 369), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 369), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 369), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 369).

6.435.3.4 `virtual Message* cms::Message::clone () const` [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 173), `activemq::commands::ActiveMQBytesMessage` (p. 200), `activemq::commands::ActiveMQMapMessage` (p. 311), `activemq::commands::ActiveMQMessage` (p. 342), `activemq::commands::ActiveMQObjectMessage` (p. 384), `activemq::commands::ActiveMQStreamMessage` (p. 467), `activemq::commands::ActiveMQTextMessage` (p. 574), and `cms::BytesMessage` (p. 941).

6.435.3.5 `virtual bool cms::Message::getBooleanProperty (const std::string & name) const throw (MessageFormatException, CMSException) [pure virtual]`

Gets a boolean property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 369), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 369), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 369).

6.435.3.6 `virtual unsigned char cms::Message::getByteProperty (const std::string & name) const throw (MessageFormatException, CMSException) [pure virtual]`

Gets a byte property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 370), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 370).

6.435.3.7 `virtual std::string cms::Message::getCMSCorrelationID () const throw (CMSEException) [pure virtual]`

Gets the correlation ID for the message. This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

Returns:

string representation of the correlation Id

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 370), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 370).

6.435.3.8 `virtual int cms::Message::getCMSDeliveryMode () const throw (CMSEException) [pure virtual]`

Gets the **DeliveryMode** (p. 1478) for this message.

Returns:

DeliveryMode (p. 1478) enumerated value.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 370), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 370), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 370).

6.435.3.9 `virtual const Destination* cms::Message::getCMSDestination () const throw (CMSEException) [pure virtual]`

Gets the **Destination** (p. 1480) object for this message. The **CMSDestination** header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its **CMSDestination** value must be equivalent to the value assigned when it was sent.

Returns:

Destination (p. 1480) object

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 371), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 371).

6.435.3.10 `virtual long long cms::Message::getCMSExpiration () const throw (CMSException) [pure virtual]`

Gets the message's expiration value. When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

Returns:

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 371), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 371).

6.435.3.11 `virtual std::string cms::Message::getCMSMessageID () const throw (CMSException) [pure virtual]`

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider. When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A `CMSMessageID` is a `String` value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All `CMSMessageID` values must start with the prefix `'ID:'`. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the `MessageProducer.setDisableMessageID` (p. 2337) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

Returns:

provider-assigned message id

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 371), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 371), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 371).

6.435.3.12 `virtual int cms::Message::getCMSPriority () const throw (CMSException) [pure virtual]`

Gets the message priority level. The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

Returns:

priority value

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 372), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 372).

6.435.3.13 virtual bool cms::Message::getCMSRedelivered () const throw (CMSEException) [pure virtual]

Gets an indication of whether this message is being redelivered. If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

Returns:

true if this message is being redelivered

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 372), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 372).

6.435.3.14 virtual const cms::Destination* cms::Message::getCMSReplyTo () const throw (CMSEException) [pure virtual]

Gets the **Destination** (p. 1480) object to which a reply to this message should be sent.

Returns:

Destination (p. 1480) to which to send a response to this message

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 372), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 372).

6.435.3.15 virtual long long cms::Message::getCMSTimestamp () const throw (CMSEException) [pure virtual]

Gets the message timestamp. The CMSTimestamp header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, CMSTimestamp is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the `MessageProducer.setDisableMessageTimestamp` method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

Returns:

the message timestamp

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 372), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 372), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 372).

6.435.3.16 `virtual std::string cms::Message::getCMSType () const throw (CMSException) [pure virtual]`

Gets the message type identifier supplied by the client when the message was sent.

Returns:

the message type

See also:

`setCMSType` (p. 2183)

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 373), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 373).

6.435.3.17 `virtual double cms::Message::getDoubleProperty (const std::string & name) const throw (MessageFormatException, CMSException) [pure virtual]`

Gets a double property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 373), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 373).

6.435.3.18 `virtual float cms::Message::getFloatProperty (const std::string & name) const throw (MessageFormatException, CMSEException) [pure virtual]`

Gets a float property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 373), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 373), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 373).

6.435.3.19 `virtual int cms::Message::getIntProperty (const std::string & name) const throw (MessageFormatException, CMSEException) [pure virtual]`

Gets a int property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 374), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 374).

6.435.3.20 `virtual long long cms::Message::getLongProperty (const std::string & name) const throw (MessageFormatException, CMSEException)` [pure virtual]

Gets a long property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 374), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 374), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 374).

6.435.3.21 `virtual std::vector<std::string> cms::Message::getPropertyNames () const throw (CMSEException)` [pure virtual]

Retrieves the property names.

Returns:

The complete set of property names currently in this message.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 375), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 375).

6.435.3.22 `virtual short cms::Message::getShortProperty (const std::string & name) const throw (MessageFormatException, CMSEException) [pure virtual]`

Gets a short property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 375), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 375).

6.435.3.23 `virtual std::string cms::Message::getStringProperty (const std::string & name) const throw (MessageFormatException, CMSEException) [pure virtual]`

Gets a string property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSEException (p. 1031) if the property does not exist.

MessageFormatException (p. 2278) - if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 375), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 375), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 375).

6.435.3.24 `virtual bool cms::Message::propertyExists (const std::string & name) const throw (CMSExcption)` [pure virtual]

Indicates whether or not a given property exists.

Parameters:

name The name of the property to look up.

Returns:

True if the property exists in this message.

Exceptions:

CMSExcption (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 376), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 376), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 376), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 376), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 376), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 376).

6.435.3.25 `virtual void cms::Message::setBooleanProperty (const std::string & name, bool value) throw (MessageNotWriteableException, CMSExcption)` [pure virtual]

Sets a boolean property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSExcption (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 376), `activemq::commands::ActiveMQMessageTemplate<`

cms::MapMessage > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::Message > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::StreamMessage > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::TextMessage > (p. 376), and **activemq::commands::ActiveMQMessageTemplate**<
cms::ObjectMessage > (p. 376).

6.435.3.26 **virtual void cms::Message::setByteProperty (const std::string & name, unsigned char value) throw (MessageNotWriteableException, CMSException) [pure virtual]**

Sets a byte property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**<
cms::BytesMessage > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::MapMessage > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::Message > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::StreamMessage > (p. 376), **activemq::commands::ActiveMQMessageTemplate**<
cms::TextMessage > (p. 376), and **activemq::commands::ActiveMQMessageTemplate**<
cms::ObjectMessage > (p. 376).

6.435.3.27 **virtual void cms::Message::setCMSCorrelationID (const std::string & correlationId) throw (CMSException) [pure virtual]**

Sets the correlation ID for the message. A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

Parameters:

correlationId The message ID of a message being referred to.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 377), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 377).

6.435.3.28 `virtual void cms::Message::setCMSDeliveryMode (int mode) throw (CMSEException)` [pure virtual]

Sets the **DeliveryMode** (p. 1478) for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

mode **DeliveryMode** (p. 1478) enumerated value.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 377), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 377).

6.435.3.29 `virtual void cms::Message::setCMSDestination (const Destination * destination) throw (CMSEException)` [pure virtual]

Sets the **Destination** (p. 1480) object for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

destination **Destination** (p. 1480) Object

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 377), `activemq::commands::ActiveMQMessageTemplate<`

cms::MapMessage > (p. 377), activemq::commands::ActiveMQMessageTemplate<
 cms::Message > (p. 377), activemq::commands::ActiveMQMessageTemplate<
 cms::StreamMessage > (p. 377), activemq::commands::ActiveMQMessageTemplate<
 cms::TextMessage > (p. 377), and activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage > (p. 377).

6.435.3.30 virtual void cms::Message::setCMSExpiration (long long *expireTime*) throw (CMSException) [pure virtual]

Sets the message's expiration value. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

expireTime the message's expiration time

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<**
 cms::BytesMessage > (p. 378), **activemq::commands::ActiveMQMessageTemplate<**
 cms::MapMessage > (p. 378), **activemq::commands::ActiveMQMessageTemplate<**
 cms::Message > (p. 378), **activemq::commands::ActiveMQMessageTemplate<**
 cms::StreamMessage > (p. 378), **activemq::commands::ActiveMQMessageTemplate<**
 cms::TextMessage > (p. 378), and **activemq::commands::ActiveMQMessageTemplate<**
 cms::ObjectMessage > (p. 378).

6.435.3.31 virtual void cms::Message::setCMSMessageID (const std::string & *id*) throw (CMSException) [pure virtual]

Sets the message ID. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

id the ID of the message

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

6.435.3.32 virtual void cms::Message::setCMSPriority (int *priority*) throw (CMSException) [pure virtual]

Sets the Priority Value for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

priority priority value for this message

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 378), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 378).

6.435.3.33 `virtual void cms::Message::setCMSRedelivered (bool redelivered) throw (CMSEException)` [pure virtual]

Specifies whether this message is being redelivered. This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

Parameters:

redelivered boolean redelivered value

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 378), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 378), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 378).

6.435.3.34 `virtual void cms::Message::setCMSReplyTo (const cms::Destination * destination) throw (CMSEException)` [pure virtual]

Sets the **Destination** (p.1480) object to which a reply to this message should be sent. The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 2674) object or a **Topic** (p. 3215) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

Parameters:

destination **Destination** (p. 1480) to which to send a response to this message

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 379), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 379).

6.435.3.35 `virtual void cms::Message::setCMSTimestamp (long long timeStamp) throw (CMSEException)` [pure virtual]

Sets the message timestamp. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

timeStamp integer time stamp value

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 379), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 379).

6.435.3.36 `virtual void cms::Message::setCMSType (const std::string & type) throw (CMSEException)` [pure virtual]

Sets the message type. Some CMS providers use a message repository that contains the definitions of messages sent by applications. The `CMSType` header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to `CMSType`, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for `CMSType` that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

Parameters:

type the message type

See also:

`getCMSType` (p. 2174)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 379), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 379), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 379).

6.435.3.37 `virtual void cms::Message::setDoubleProperty (const std::string & name, double value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a double property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 380), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 380).

6.435.3.38 `virtual void cms::Message::setFloatProperty (const std::string & name, float value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a float property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 380), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 380).

6.435.3.39 `virtual void cms::Message::setIntProperty (const std::string & name, int value) throw (MessageNotWriteableException, CMSEException)` [pure virtual]

Sets a int property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::Message >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >` (p. 380), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >` (p. 380), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >` (p. 380).

6.435.3.40 `virtual void cms::Message::setLongProperty (const std::string & name, long long value) throw (MessageNotWriteableException, CMSEException)` [pure virtual]

Sets a long property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >` (p. 381), `activemq::commands::ActiveMQMessageTemplate<`

cms::MapMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::Message > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::StreamMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::TextMessage > (p. 381), and **activemq::commands::ActiveMQMessageTemplate**<
cms::ObjectMessage > (p. 381).

6.435.3.41 **virtual void cms::Message::setShortProperty (const std::string & name, short value) throw (MessageNotWriteableException, CMSEException)** [pure virtual]

Sets a short property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**<
cms::BytesMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::MapMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::Message > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::StreamMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::TextMessage > (p. 381), and **activemq::commands::ActiveMQMessageTemplate**<
cms::ObjectMessage > (p. 381).

6.435.3.42 **virtual void cms::Message::setStringProperty (const std::string & name, const std::string & value) throw (MessageNotWriteableException, CMSEException)** [pure virtual]

Sets a string property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSEException (p. 1031) - if the name is an empty string.

MessageNotWriteableException (p. 2331) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**<
cms::BytesMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::MapMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::Message > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::StreamMessage > (p. 381), **activemq::commands::ActiveMQMessageTemplate**<
cms::TextMessage > (p. 381), and **activemq::commands::ActiveMQMessageTemplate**<
cms::ObjectMessage > (p. 381).

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

6.436 activemq::commands::MessageAck Class Reference

#include <src/main/activemq/commands/MessageAck.h> Inheritance diagram for activemq::commands::MessageAck:

Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char ackType)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &firstMessageId)
- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &lastMessageId)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int messageCount)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEACK** = 22

Protected Member Functions

- **MessageAck** (const MessageAck &)
- **MessageAck & operator=** (const MessageAck &)

Protected Attributes

- **Pointer< ActiveMQDestination > destination**
- **Pointer< TransactionId > transactionId**
- **Pointer< ConsumerId > consumerId**
- unsigned char **ackType**
- **Pointer< MessageId > firstMessageId**
- **Pointer< MessageId > lastMessageId**
- int **messageCount**

6.436.1 Constructor & Destructor Documentation

6.436.1.1 **activemq::commands::MessageAck::MessageAck** (const MessageAck &)
[inline, protected]

6.436.1.2 **activemq::commands::MessageAck::MessageAck** ()

6.436.1.3 **virtual activemq::commands::MessageAck::~~MessageAck** () [virtual]

6.436.2 Member Function Documentation

6.436.2.1 **virtual MessageAck* activemq::commands::MessageAck::cloneDataStructure** ()
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.436.2.2 **virtual void activemq::commands::MessageAck::copyDataStructure**
(const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.436.2.3 `virtual bool activemq::commands::MessageAck::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.436.2.4 `virtual unsigned char activemq::commands::MessageAck::getAckType () const` [virtual]

6.436.2.5 `virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ()` [virtual]

6.436.2.6 `virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId () const` [virtual]

6.436.2.7 `virtual unsigned char activemq::commands::MessageAck::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements `activemq::commands::DataStructure` (p. 1464).

- 6.436.2.8** `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ()`
[virtual]
- 6.436.2.9** `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () const` [virtual]
- 6.436.2.10** `virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ()`
[virtual]
- 6.436.2.11** `virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ()`
`const` [virtual]
- 6.436.2.12** `virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ()`
[virtual]
- 6.436.2.13** `virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ()`
`const` [virtual]
- 6.436.2.14** `virtual int activemq::commands::MessageAck::getMessageCount () const`
[virtual]
- 6.436.2.15** `virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId ()`
[virtual]
- 6.436.2.16** `virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () const`
[virtual]
- 6.436.2.17** `virtual bool activemq::commands::MessageAck::isMessageAck () const`
[inline, virtual]

Returns:

an answer of true to the `isMessageAck()` (p. 2191) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 653).

- 6.436.2.18 `MessageAck& activemq::commands::MessageAck::operator= (const MessageAck &) [inline, protected]`
- 6.436.2.19 `virtual void activemq::commands::MessageAck::setAckType (unsigned char ackType) [virtual]`
- 6.436.2.20 `virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.436.2.21 `virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.436.2.22 `virtual void activemq::commands::MessageAck::setFirstMessageId (const Pointer< MessageId > & firstMessageId) [virtual]`
- 6.436.2.23 `virtual void activemq::commands::MessageAck::setLastMessageId (const Pointer< MessageId > & lastMessageId) [virtual]`
- 6.436.2.24 `virtual void activemq::commands::MessageAck::setMessageCount (int messageCount) [virtual]`
- 6.436.2.25 `virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.436.2.26 `virtual std::string activemq::commands::MessageAck::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

- 6.436.2.27 `virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.1067).

6.436.3 Field Documentation

- 6.436.3.1 `unsigned char activemq::commands::MessageAck::ackType` [protected]
- 6.436.3.2 `Pointer<ConsumerId> activemq::commands::MessageAck::consumerId` [protected]
- 6.436.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination` [protected]
- 6.436.3.4 `Pointer<MessageId> activemq::commands::MessageAck::firstMessageId` [protected]
- 6.436.3.5 `const unsigned char activemq::commands::MessageAck::ID_ - MESSAGEACK = 22` [static]
- 6.436.3.6 `Pointer<MessageId> activemq::commands::MessageAck::lastMessageId` [protected]
- 6.436.3.7 `int activemq::commands::MessageAck::messageCount` [protected]
- 6.436.3.8 `Pointer<TransactionId> activemq::commands::MessageAck::transactionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageAck.h`

6.437 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2194).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.437.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2194). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.437.2 Constructor & Destructor Documentation

6.437.2.1 `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshaller()` [inline]

6.437.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

6.437.3 Member Function Documentation

6.437.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.437.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.437.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.437.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.437.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.437.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.437.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h

6.438 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2198).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.438.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2198). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.438.2 Constructor & Destructor Documentation

6.438.2.1 `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::MessageAckMarshaller()` `[inline]`

6.438.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::~~MessageAckMarshaller()` `[inline, virtual]`

6.438.3 Member Function Documentation

6.438.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.438.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.438.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.438.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.438.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.438.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.438.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h

6.439 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2202).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.439.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2202). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.439.2 Constructor & Destructor Documentation

6.439.2.1 `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshaller()` [inline]

6.439.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

6.439.3 Member Function Documentation

6.439.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.439.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.439.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.439.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.439.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.439.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.439.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h

6.440 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2206).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.440.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2206). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.440.2 Constructor & Destructor Documentation

6.440.2.1 `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::MessageAckMarshaller()` `[inline]`

6.440.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::~~MessageAckMarshaller()` `[inline, virtual]`

6.440.3 Member Function Documentation

6.440.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.440.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.440.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.440.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.440.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.440.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.440.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h

6.441 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2210).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.441.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2210). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.441.2 Constructor & Destructor Documentation

6.441.2.1 `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshaller()` [inline]

6.441.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~~MessageAckMarshaller()` [inline, virtual]

6.441.3 Member Function Documentation

6.441.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.441.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.441.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.441.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.441.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.441.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.441.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h

6.442 cms::MessageConsumer Class Reference

A client uses a `MessageConsumer` (p. 2214) to received messages from a destination.

```
#include <src/main/cms/MessageConsumer.h> Inheritance      diagram      for
cms::MessageConsumer:
```

Public Member Functions

- virtual `~MessageConsumer ()`
- virtual `Message * receive ()=0 throw (CMSEException)`
*Synchronously Receive a **Message** (p. 2163).*
- virtual `Message * receive (int millisecs)=0 throw (CMSEException)`
*Synchronously Receive a **Message** (p. 2163), time out after defined interval.*
- virtual `Message * receiveNoWait ()=0 throw (CMSEException)`
*Receive a **Message** (p. 2163), does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void `setMessageListener (MessageListener *listener)=0 throw (CMSEException)`
*Sets the **MessageListener** (p. 2304) that this class will send notifs on.*
- virtual `MessageListener * getMessageListener () const =0 throw (CMSEException)`
*Gets the **MessageListener** (p. 2304) that this class will send new **Message** (p. 2163) notification events to.*
- virtual std::string `getMessageSelector () const =0 throw (cms::CMSEException)`
Gets this message consumer's message selector expression.

6.442.1 Detailed Description

A client uses a `MessageConsumer` (p. 2214) to received messages from a destination. A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its `receive` methods. There are several variations of `receive` that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a `MessageListener` (p. 2304) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the `MessageListener` (p. 2304)'s `onMessage` method.

When the `MessageConsumer`'s close method is called the method can block while an asynchronous message delivery is in progress or until a `receive` operation completes. A blocked consumer in a `receive` call will return a Null when the close method is called.

See also:

[MessageListener](#) (p. 2304)

Since:

1.0

6.442.2 Constructor & Destructor Documentation

6.442.2.1 `virtual cms::MessageConsumer::~MessageConsumer () [inline, virtual]`

6.442.3 Member Function Documentation

6.442.3.1 `virtual MessageListener* cms::MessageConsumer::getMessageListener () const throw (CMSEException) [pure virtual]`

Gets the [MessageListener](#) (p. 2304) that this class will send new [Message](#) (p. 2163) notification events to.

Returns:

The listener of messages received by this consumer

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in [activemq::cmsutil::CachedConsumer](#) (p. 954), and [activemq::core::ActiveMQConsumer](#) (p. 268).

6.442.3.2 `virtual std::string cms::MessageConsumer::getMessageSelector () const throw (cms::CMSEException) [pure virtual]`

Gets this message consumer's message selector expression.

Returns:

This Consumer's selector expression or "".

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in [activemq::cmsutil::CachedConsumer](#) (p. 954), and [activemq::core::ActiveMQConsumer](#) (p. 268).

6.442.3.3 `virtual Message* cms::MessageConsumer::receive (int millisecs) throw (CMSEException) [pure virtual]`

Synchronously Receive a [Message](#) (p. 2163), time out after defined interval. Returns null if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 954), and `activemq::core::ActiveMQConsumer` (p. 269).

6.442.3.4 `virtual Message* cms::MessageConsumer::receive () throw (CMSEException) [pure virtual]`

Synchronously Receive a **Message** (p. 2163).

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 955), and `activemq::core::ActiveMQConsumer` (p. 269).

6.442.3.5 `virtual Message* cms::MessageConsumer::receiveNoWait () throw (CMSEException) [pure virtual]`

Receive a **Message** (p. 2163), does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 955), and `activemq::core::ActiveMQConsumer` (p. 269).

6.442.3.6 `virtual void cms::MessageConsumer::setMessageListener (MessageListener * listener) throw (CMSEException) [pure virtual]`

Sets the **MessageListener** (p. 2304) that this class will send notifs on.

Parameters:

listener The listener of messages received by this consumer.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 955), and **activemq::core::ActiveMQConsumer** (p. 270).

The documentation for this class was generated from the following file:

- src/main/cms/MessageConsumer.h

6.443 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the CmsTemplate (p. 1041).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```

Public Member Functions

- virtual `~MessageCreator()`
- virtual `cms::Message * createMessage (cms::Session *session)=0` throw (`cms::CMSEException`)

Creates a message from the given session.

6.443.1 Detailed Description

Creates the user-defined message to be sent by the CmsTemplate (p. 1041).

6.443.2 Constructor & Destructor Documentation

- 6.443.2.1 virtual `activemq::cmsutil::MessageCreator::~MessageCreator()`
[inline, virtual]

6.443.3 Member Function Documentation

- 6.443.3.1 virtual `cms::Message* activemq::cmsutil::MessageCreator::createMessage (cms::Session * session)` throw (`cms::CMSEException`) [pure virtual]

Creates a message from the given session.

Parameters:

session the CMS Session

Exceptions:

cms::CMSEException (p. 1031) if thrown by CMS API methods

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/MessageCreator.h`

6.444 activemq::commands::MessageDispatch Class Reference

#include <src/main/activemq/commands/MessageDispatch.h> Inheritance diagram for activemq::commands::MessageDispatch:

Public Member Functions

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageDispatch** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE_DISPATCH** = 21

Protected Member Functions

- `MessageDispatch (const MessageDispatch &)`
- `MessageDispatch & operator= (const MessageDispatch &)`

Protected Attributes

- `Pointer< ConsumerId > consumerId`
- `Pointer< ActiveMQDestination > destination`
- `Pointer< Message > message`
- `int redeliveryCounter`

6.444.1 Constructor & Destructor Documentation

- 6.444.1.1** `activemq::commands::MessageDispatch::MessageDispatch (const MessageDispatch &) [inline, protected]`
- 6.444.1.2** `activemq::commands::MessageDispatch::MessageDispatch ()`
- 6.444.1.3** `virtual activemq::commands::MessageDispatch::~~MessageDispatch () [virtual]`

6.444.2 Member Function Documentation

- 6.444.2.1** `virtual MessageDispatch* activemq::commands::MessageDispatch::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

- 6.444.2.2** `virtual void activemq::commands::MessageDispatch::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.444.2.3 virtual bool activemq::commands::MessageDispatch::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.444.2.4 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () [virtual]

6.444.2.5 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () const [virtual]

6.444.2.6 virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.444.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ()`
[virtual]
- 6.444.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination () const`
[virtual]
- 6.444.2.9 `virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage ()`
[virtual]
- 6.444.2.10 `virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage () const` [virtual]
- 6.444.2.11 `virtual int activemq::commands::MessageDispatch::getRedeliveryCounter () const`
[virtual]
- 6.444.2.12 `virtual bool activemq::commands::MessageDispatch::isMessageDispatch () const` [inline, virtual]

Returns:

an answer of true to the `isMessageDispatch()` (p. 2222) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 653).

- 6.444.2.13 `MessageDispatch& activemq::commands::MessageDispatch::operator=(const MessageDispatch &)` [inline, protected]
- 6.444.2.14 `virtual void activemq::commands::MessageDispatch::setConsumerId(const Pointer< ConsumerId > & consumerId)` [virtual]
- 6.444.2.15 `virtual void activemq::commands::MessageDispatch::setDestination(const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.444.2.16 `virtual void activemq::commands::MessageDispatch::setMessage(const Pointer< Message > & message)` [virtual]
- 6.444.2.17 `virtual void activemq::commands::MessageDispatch::setRedeliveryCounter(int redeliveryCounter)` [virtual]
- 6.444.2.18 `virtual std::string activemq::commands::MessageDispatch::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

6.444.2.19 `virtual Pointer<Command> activemq::commands::MessageDispatch::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.444.3 Field Documentation

6.444.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId [protected]`

6.444.3.2 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination [protected]`

6.444.3.3 `const unsigned char activemq::commands::MessageDispatch::ID_ - MESSAGEDISPATCH = 21 [static]`

6.444.3.4 `Pointer<Message> activemq::commands::MessageDispatch::message [protected]`

6.444.3.5 `int activemq::commands::MessageDispatch::redeliveryCounter [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatch.h`

6.445 activemq::core::MessageDispatchChannel Class Reference

#include <src/main/activemq/core/MessageDispatchChannel.h> Inheritance diagram for activemq::core::MessageDispatchChannel:

Public Member Functions

- **MessageDispatchChannel** ()
- virtual **~MessageDispatchChannel** ()
- void **enqueue** (const **Pointer**< **MessageDispatch** > &message)
Add a Message to the Channel behind all pending message.
- void **enqueueFirst** (const **Pointer**< **MessageDispatch** > &message)
Add a message to the front of the Channel.
- bool **isEmpty** () const
- bool **isClosed** () const
- bool **isRunning** () const
- **Pointer**< **MessageDispatch** > **dequeue** (long long timeout) throw (exceptions::ActiveMQException)
Used to get an enqueued message.
- **Pointer**< **MessageDispatch** > **dequeueNoWait** ()
Used to get an enqueued message if there is one queued right now.
- **Pointer**< **MessageDispatch** > **peek** () const
Peek in the Queue and return the first message in the Channel without removing it from the channel.
- void **start** ()
Starts dispatch of messages from the Channel.
- void **stop** ()
Stops dispatch of message from the Channel.
- void **close** ()
Close this channel no messages will be dispatched after this method is called.
- void **clear** ()
Clear the Channel, all pending messages are removed.
- int **size** () const
- std::vector< **Pointer**< **MessageDispatch** > > **removeAll** ()
Remove all messages that are currently in the Channel and return them as a list of Messages.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.445.1 Constructor & Destructor Documentation

6.445.1.1 activemq::core::MessageDispatchChannel::MessageDispatchChannel ()

6.445.1.2 virtual activemq::core::MessageDispatchChannel::~~MessageDispatchChannel () [virtual]

6.445.2 Member Function Documentation

6.445.2.1 void activemq::core::MessageDispatchChannel::clear ()

Clear the Channel, all pending messages are removed.

6.445.2.2 void activemq::core::MessageDispatchChannel::close ()

Close this channel no messages will be dispatched after this method is called.

6.445.2.3 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue (long long timeout) throw (exceptions::ActiveMQException)`

Used to get an enqueued message. The amount of time this method blocks is based on the timeout value. - if `timeout==1` then it blocks until a message is received. - if `timeout==0` then it tries to not block at all, it returns a message if it is available - if `timeout>0` then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

Returns:

null if we timeout or if the consumer is closed.

Exceptions:

ActiveMQException

6.445.2.4 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ()`

Used to get an enqueued message if there is one queued right now. If there is no waiting message then this method returns Null.

Returns:

a message if there is one in the queue.

6.445.2.5 `void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & message)`

Add a Message to the Channel behind all pending message.

Parameters:

message - The message to add to the Channel.

6.445.2.6 `void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & message)`

Add a message to the front of the Channel.

Parameters:

message - The Message to add to the front of the Channel.

6.445.2.7 `bool activemq::core::MessageDispatchChannel::isClosed () const [inline]`

Returns:

has the Queue been closed.

6.445.2.8 `bool activemq::core::MessageDispatchChannel::isEmpty () const`**Returns:**

true if there are no messages in the Channel.

6.445.2.9 `bool activemq::core::MessageDispatchChannel::isRunning () const`
[inline]**Returns:**

true if the Channel currently running and will dequeue message.

6.445.2.10 `virtual void activemq::core::MessageDispatchChannel::lock () throw (`
`decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.445.2.11 `virtual void activemq::core::MessageDispatchChannel::notify`
`() throw (decaf::lang::exceptions::RuntimeException,`
`decaf::lang::exceptions::IllegalMonitorStateException)` [inline,
virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

6.445.2.12 `virtual void activemq::core::MessageDispatchChannel::notifyAll`
`() throw (decaf::lang::exceptions::RuntimeException,`
`decaf::lang::exceptions::IllegalMonitorStateException)` [inline,
virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3125).

6.445.2.13 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek () const`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns:

a message if there is one in the queue.

6.445.2.14 `std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ()`

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns:

a list of Messages that was previously in the Channel.

6.445.2.15 `int activemq::core::MessageDispatchChannel::size () const`

Returns:

the number of Messages currently in the Channel.

6.445.2.16 `void activemq::core::MessageDispatchChannel::start ()`

Starts dispatch of messages from the Channel.

6.445.2.17 `void activemq::core::MessageDispatchChannel::stop ()`

Stops dispatch of message from the Channel.

6.445.2.18 `virtual bool activemq::core::MessageDispatchChannel::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.445.2.19 `virtual void activemq::core::MessageDispatchChannel::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.445.2.20 `virtual void activemq::core::MessageDispatchChannel::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.445.2.21 `virtual void activemq::core::MessageDispatchChannel::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

```
6.445.2.22  virtual void activemq::core::MessageDispatchChannel::wait  
            () throw ( decaf::lang::exceptions::RuntimeException,  
                      decaf::lang::exceptions::IllegalMonitorStateException,  
                      decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- src/main/activemq/core/MessageDispatchChannel.h

6.446 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2231).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.446.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2231). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.446.2 Constructor & Destructor Documentation

6.446.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispatch`
 () [inline]

6.446.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~~MessageDispatch`
 () [inline, virtual]

6.446.3 Member Function Documentation

6.446.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject`
 () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.446.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructure`
 () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.446.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal`
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.446.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.446.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.446.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.446.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h

6.447 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2235).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.447.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2235). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.447.2 Constructor & Destructor Documentation

6.447.2.1 `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::MessageDispatch()` [inline]

6.447.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::~~MessageDispatch()` [inline, virtual]

6.447.3 Member Function Documentation

6.447.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.447.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.447.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.447.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.447.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.447.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.447.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h

6.448 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2239).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.448.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2239). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.448.2 Constructor & Destructor Documentation

6.448.2.1 `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispatch`
 () [inline]

6.448.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~~MessageDispatch`
 () [inline, virtual]

6.448.3 Member Function Documentation

6.448.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject`
 () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.448.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructure`
 () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.448.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal`
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.448.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.448.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.448.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.448.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h

6.449 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2243).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.449.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2243). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.449.2 Constructor & Destructor Documentation

6.449.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispatch()` [inline]

6.449.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~~MessageDispatch()` [inline, virtual]

6.449.3 Member Function Documentation

6.449.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.449.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.449.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.449.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.449.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.449.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.449.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h

6.450 activemq::wireformat::openwire::marshal::v5::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2247).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.450.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2247). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.450.2 Constructor & Destructor Documentation

6.450.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::MessageDispatch`
 () [inline]

6.450.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::~~MessageDispatch`
 () [inline, virtual]

6.450.3 Member Function Documentation

6.450.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::createObject`
 () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.450.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::getDataStructure`
 () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.450.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseMarshal`
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.450.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.450.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.450.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.450.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h

6.451 activemq::commands::MessageDispatchNotification Class Reference

#include <src/main/activemq/commands/MessageDispatchNotification.h> Inheritance diagram for activemq::commands::MessageDispatchNotification:

Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageDispatchNotification * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long deliverySequenceId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE_DISPATCH_NOTIFICATION** = 90

Protected Member Functions

- **MessageDispatchNotification** (const MessageDispatchNotification &)
- **MessageDispatchNotification & operator=** (const MessageDispatchNotification &)

Protected Attributes

- **Pointer< ConsumerId > consumerId**
- **Pointer< ActiveMQDestination > destination**
- **long long deliverySequenceId**
- **Pointer< MessageId > messageId**

6.451.1 Constructor & Destructor Documentation

- 6.451.1.1** **activemq::commands::MessageDispatchNotification::MessageDispatchNotification** (const MessageDispatchNotification &) [inline, protected]
- 6.451.1.2** **activemq::commands::MessageDispatchNotification::MessageDispatchNotification** ()
- 6.451.1.3** **virtual**
activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification () [virtual]

6.451.2 Member Function Documentation

- 6.451.2.1** **virtual MessageDispatchNotification* ac-**
tivemq::commands::MessageDispatchNotification::cloneDataStructure ()
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

- 6.451.2.2** **virtual void ac-**
tivemq::commands::MessageDispatchNotification::copyDataStructure
(const DataStructure * src) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.451.2.3 virtual bool activemq::commands::MessageDispatchNotification::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.451.2.4 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId () [virtual]

6.451.2.5 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId () const [virtual]

6.451.2.6 virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.451.2.7 `virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceId () const [virtual]`
- 6.451.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination () [virtual]`
- 6.451.2.9 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination () const [virtual]`
- 6.451.2.10 `virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId () [virtual]`
- 6.451.2.11 `virtual const Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId () const [virtual]`
- 6.451.2.12 `virtual bool activemq::commands::MessageDispatchNotification::isMessageDispatchNotification () const [inline, virtual]`

Returns:

an answer of true to the `isMessageDispatchNotification()` (p. 2254) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 654).

- 6.451.2.13** `MessageDispatchNotification& activemq::commands::MessageDispatchNotification::operator= (const MessageDispatchNotification &) [inline, protected]`
- 6.451.2.14** `virtual void activemq::commands::MessageDispatchNotification::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.451.2.15** `virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceId (long long deliverySequenceId) [virtual]`
- 6.451.2.16** `virtual void activemq::commands::MessageDispatchNotification::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.451.2.17** `virtual void activemq::commands::MessageDispatchNotification::setMessageId (const Pointer< MessageId > & messageId) [virtual]`
- 6.451.2.18** `virtual std::string activemq::commands::MessageDispatchNotification::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

- 6.451.2.19** `virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1067).

6.451.3 Field Documentation

- 6.451.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId`
[protected]
- 6.451.3.2 `long long activemq::commands::MessageDispatchNotification::deliverySequenceId`
[protected]
- 6.451.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination`
[protected]
- 6.451.3.4 `const unsigned char activemq::commands::MessageDispatchNotification::ID - MESSAGEDISPATCHNOTIFICATION = 90` [static]
- 6.451.3.5 `Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatchNotification.h`

6.452 ac-

tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller

Class Reference

6.452 ²²⁶¹activemq::wireformat::openwire::marshal::v2::MessageDispatchNo

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2257).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller
diagram for activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.452.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2257). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.452.2 Constructor & Destructor Documentation

6.452.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

6.452.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

6.452.3 Member Function Documentation

6.452.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.452.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.452.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.452 ac-
ativemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
Class Reference **2263**
 Reimplemented from **ativemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
 (p. 686).

6.452.3.4 virtual void ac-
ativemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseU
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (**
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **ativemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
 (p. 687).

6.452.3.5 virtual int ac-
ativemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **ativemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
 (p. 688).

6.452.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.452.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h

6.453 ac-

tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller

Class Reference

6.453 ²²⁶⁵activemq::wireformat::openwire::marshal::v4::MessageDispatchNo

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 2261).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller
diagram for activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.453.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 2261). NOTE!: This file is auto generated - do not modify! if you need to make a change,
please see the Java Classes in the activemq-openwire-generator module

6.453.2 Constructor & Destructor Documentation

6.453.2.1 `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

6.453.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

6.453.3 Member Function Documentation

6.453.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.453.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.453.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.453 **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**
Class Reference 2267
 Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
 (p. 672).

6.453.3.4 **virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException)** [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
 (p. 673).

6.453.3.5 **virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)** [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
 (p. 674).

6.453.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.453.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2265).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller
 diagram for activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.454.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2265). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.454.2 Constructor & Destructor Documentation

6.454.2.1 `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

6.454.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

6.454.3 Member Function Documentation

6.454.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.454.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.454.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.454 **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**
Class Reference **2271**
 Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
 (p. 658).

6.454.3.4 **virtual void** **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseUnmarshal**
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
 (p. 659).

6.454.3.5 **virtual int** **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightMarshal**
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
 (p. 660).

6.454.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 661).

6.454.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 662).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h`

6.455 ac-

tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller

Class Reference

6.455 — activemq::wireformat::openwire::marshal::v1::MessageDispatchNo²²⁷³

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2269).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller
diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.455.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2269). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.455.2 Constructor & Destructor Documentation

6.455.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

6.455.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

6.455.3 Member Function Documentation

6.455.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.455.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.455.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::marshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.455 ac-
ativemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
Class Reference **2275**
 Reimplemented from **ativemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
 (p. 665).

6.455.3.4 virtual void ac-
ativemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseU
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (**
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **ativemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
 (p. 666).

6.455.3.5 virtual int ac-
ativemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **ativemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
 (p. 667).

6.455.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 668).

6.455.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 669).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h`

6.456 ac-

tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller

Class Reference

6.456 — activemq::wireformat::openwire::marshal::v5::MessageDispatchNo²²⁷⁷

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2273).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller
diagram for activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn,
utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (de-
caf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut,
utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **com-
mands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut)
throw (decaf::io::IOException)
Write a object instance to data output stream.

6.456.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2273). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.456.2 Constructor & Destructor Documentation

6.456.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller()` [inline]

6.456.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller()` [inline, virtual]

6.456.3 Member Function Documentation

6.456.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::createDataStructure()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.456.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.456.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

6.456 **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**
Class Reference **2279**
 Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
 (p. 679).

6.456.3.4 **virtual void** **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseUnmarshal**
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
 (p. 680).

6.456.3.5 **virtual int** **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightMarshal**
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
 (p. 681).

6.456.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.456.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h

6.457 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3081) or **BytesMessage** (p. 938) is being read.

#include <src/main/cms/MessageEOFException.h> Inheritance diagram for cms::MessageEOFException:

Public Member Functions

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

6.457.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3081) or **BytesMessage** (p. 938) is being read.

Since:

1.3

6.457.2 Constructor & Destructor Documentation

- 6.457.2.1 **cms::MessageEOFException::MessageEOFException** () throw ()
- 6.457.2.2 **cms::MessageEOFException::MessageEOFException** (const **MessageEOFException** & *ex*) throw ()
- 6.457.2.3 **cms::MessageEOFException::MessageEOFException** (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.457.2.4 **cms::MessageEOFException::MessageEOFException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.457.2.5 **virtual cms::MessageEOFException::~~MessageEOFException** () throw ()
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageEOFException.h

6.458 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

#include <src/main/cms/MessageFormatException.h> Inheritance diagram for cms::MessageFormatException:

Public Member Functions

- **MessageFormatException** () throw ()
- **MessageFormatException** (const **MessageFormatException** &ex) throw ()
- **MessageFormatException** (const std::string &message, const std::exception *cause) throw ()
- **MessageFormatException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageFormatException** () throw ()

6.458.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type. It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if **StreamMessage.readShort** (p. 3088) is used to read a boolean value.

Since:

1.3

6.458.2 Constructor & Destructor Documentation

- 6.458.2.1 **cms::MessageFormatException::MessageFormatException** () throw ()
- 6.458.2.2 **cms::MessageFormatException::MessageFormatException** (const **MessageFormatException** & *ex*) throw ()
- 6.458.2.3 **cms::MessageFormatException::MessageFormatException** (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.458.2.4 **cms::MessageFormatException::MessageFormatException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.458.2.5 virtual **cms::MessageFormatException::~MessageFormatException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageFormatException.h

6.459 activemq::commands::MessageId Class Reference

#include <src/main/activemq/commands/MessageId.h> Inheritance diagram for activemq::commands::MessageId:

Public Types

- typedef **decaf::lang::PointerComparator**< MessageId > **COMPARATOR**

Public Member Functions

- **MessageId** ()
- **MessageId** (const **MessageId** &other)
- virtual **~MessageId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual long long **getProducerSequenceId** () const
- virtual void **setProducerSequenceId** (long long producerSequenceId)
- virtual long long **getBrokerSequenceId** () const
- virtual void **setBrokerSequenceId** (long long brokerSequenceId)
- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const
- virtual bool **operator==** (const **MessageId** &value) const
- virtual bool **operator<** (const **MessageId** &value) const
- **MessageId** & **operator=** (const **MessageId** &other)

Static Public Attributes

- static const unsigned char **ID_MESSAGEID** = 110

Protected Attributes

- **Pointer< ProducerId > producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

6.459.1 Member Typedef Documentation

6.459.1.1 `typedef decaf::lang::PointerComparator<MessageId>
activemq::commands::MessageId::COMPARATOR`

6.459.2 Constructor & Destructor Documentation

6.459.2.1 `activemq::commands::MessageId::MessageId ()`

6.459.2.2 `activemq::commands::MessageId::MessageId (const MessageId & other)`

6.459.2.3 `virtual activemq::commands::MessageId::~~MessageId () [virtual]`

6.459.3 Member Function Documentation

6.459.3.1 `virtual MessageId* activemq::commands::MessageId::cloneDataStructure
() const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.459.3.2 `virtual int activemq::commands::MessageId::compareTo (const MessageId
& value) const [virtual]`

6.459.3.3 `virtual void activemq::commands::MessageId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.459.3.4 `virtual bool activemq::commands::MessageId::equals (const MessageId & value) const [virtual]`

6.459.3.5 `virtual bool activemq::commands::MessageId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.459.3.6 `virtual long long activemq::commands::MessageId::getBrokerSequenceId () const [virtual]`

6.459.3.7 `virtual unsigned char activemq::commands::MessageId::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.459.3.8 `virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ()`
[virtual]
- 6.459.3.9 `virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId () const`
[virtual]
- 6.459.3.10 `virtual long long activemq::commands::MessageId::getProducerSequenceId ()`
`const` [virtual]
- 6.459.3.11 `virtual bool activemq::commands::MessageId::operator< (const MessageId & value) const` [virtual]
- 6.459.3.12 `MessageId& activemq::commands::MessageId::operator= (const MessageId & other)`
- 6.459.3.13 `virtual bool activemq::commands::MessageId::operator== (const MessageId & value) const` [virtual]
- 6.459.3.14 `virtual void activemq::commands::MessageId::setBrokerSequenceId (long long brokerSequenceId)` [virtual]
- 6.459.3.15 `virtual void activemq::commands::MessageId::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.459.3.16 `virtual void activemq::commands::MessageId::setProducerSequenceId (long long producerSequenceId)` [virtual]
- 6.459.3.17 `virtual std::string activemq::commands::MessageId::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.718).

6.459.4 Field Documentation

- 6.459.4.1 `long long activemq::commands::MessageId::brokerSequenceId`
[protected]
- 6.459.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID`
`= 110` [static]
- 6.459.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId`
[protected]
- 6.459.4.4 `long long activemq::commands::MessageId::producerSequenceId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

6.460 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2284).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.460.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2284). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.460.2 Constructor & Destructor Documentation

6.460.2.1 `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller()` [inline]

6.460.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

6.460.3 Member Function Documentation

6.460.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.460.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.460.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.460.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.460.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.460.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.460.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h

6.461 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2288).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.461.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2288). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.461.2 Constructor & Destructor Documentation

6.461.2.1 `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::MessageIdMarshaller()` [inline]

6.461.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

6.461.3 Member Function Documentation

6.461.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.461.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.461.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.461.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.461.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.461.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.461.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h

6.462 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2292).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.462.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2292). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.462.2 Constructor & Destructor Documentation

6.462.2.1 `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller()` [inline]

6.462.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

6.462.3 Member Function Documentation

6.462.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.462.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.462.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.462.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.462.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.462.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.462.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h

6.463 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2296).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.463.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2296). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.463.2 Constructor & Destructor Documentation

6.463.2.1 `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::MessageIdMarshaller()` [inline]

6.463.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

6.463.3 Member Function Documentation

6.463.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.463.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.463.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.463.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.463.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.463.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.463.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h

6.464 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2300).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.464.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2300). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.464.2 Constructor & Destructor Documentation

6.464.2.1 `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller()` [inline]

6.464.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~~MessageIdMarshaller()` [inline, virtual]

6.464.3 Member Function Documentation

6.464.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.464.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.464.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.464.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.464.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.464.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.464.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h

6.465 cms::MessageListener Class Reference

A `MessageListener` (p. 2304) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

Public Member Functions

- virtual `~MessageListener ()`
- virtual void `onMessage (const Message *message)=0`

*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2163) types.*

6.465.1 Detailed Description

A `MessageListener` (p. 2304) object is used to receive asynchronously delivered messages.

Since:

1.0

6.465.2 Constructor & Destructor Documentation

6.465.2.1 virtual `cms::MessageListener::~~MessageListener ()` [inline, virtual]

6.465.3 Member Function Documentation

6.465.3.1 virtual void `cms::MessageListener::onMessage (const Message * message)`
[pure virtual]

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2163) types. a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the `onMessage` function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this **Message** (p. 2163).

It is considered a programming error for this method to throw an exception.

Parameters:

message **Message** (p. 2163) object {const} pointer recipient does not own.

The documentation for this class was generated from the following file:

- `src/main/cms/MessageListener.h`

6.466 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2305).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.466.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2305). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.466.2 Constructor & Destructor Documentation

6.466.2.1 `activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller()` [inline]

6.466.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

6.466.3 Member Function Documentation

6.466.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 194), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 339), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 496), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 595).

6.466.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 195), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 340), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 363), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 497), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 596).

```
6.466.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1
          (OpenWireFormat * wireFormat, commands::DataStructure *
           dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 195), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 340), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 363), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 497), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 596).

6.466.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 195), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 231), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 340), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 363), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 497), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 596).

6.466.3.5 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 232), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 341), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 364), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 498), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 597).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h`

6.467 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2310).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.467.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2310). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.467.2 Constructor & Destructor Documentation

- 6.467.2.1 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller::MessageMarshaller()` [inline]
- 6.467.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

6.467.3 Member Function Documentation

- 6.467.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 186), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 488), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 587).

- 6.467.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 187), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 332), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 489), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 588).

```
6.467.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal1
            (OpenWireFormat * wireFormat,  commands::DataStructure *
            dataStructure,  utils::BooleanStream * bs) throw ( decaf::io::IOException
            ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 187), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 332), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 489), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 588).

6.467.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 332), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 355), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 396), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 489), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 588).

6.467.3.5 virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 224), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 333), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 356), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 397), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 490), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 589).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h`

6.468 activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2315).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.468.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2315). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.468.2 Constructor & Destructor Documentation

6.468.2.1 `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller()` [inline]

6.468.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

6.468.3 Member Function Documentation

6.468.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 178), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 323), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 387), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 480), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 579).

6.468.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 179), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 324), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 481), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 580).

```
6.468.3.3 virtual int ac-
      tivemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1
      (OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
      ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 179), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 324), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 481), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 580).

6.468.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 661).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 179), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 215), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 324), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 481), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 580).

6.468.3.5 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 662).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 180), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 216), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 348), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 482), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 581).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h`

6.469 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2320).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.469.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2320). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.469.2 Constructor & Destructor Documentation

- 6.469.2.1 `activemq::wireformat::openwire::marshal::v5::MessageMarshaller::MessageMarshaller()` [inline]
- 6.469.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

6.469.3 Member Function Documentation

- 6.469.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 190), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 226), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 335), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 399), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 492), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 591).

- 6.469.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 191), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 336), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 359), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 493), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 592).

```
6.469.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal1
            (OpenWireFormat * wireFormat,  commands::DataStructure *
            dataStructure,  utils::BooleanStream * bs) throw ( decaf::io::IOException
            ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 191), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 336), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 359), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 493), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 592).

6.469.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 191), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 336), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 359), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 400), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 493), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 592).

6.469.3.5 virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 228), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 337), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 360), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 494), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 593).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h`

6.470 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2325).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.470.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2325). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.470.2 Constructor & Destructor Documentation

6.470.2.1 `activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller()` [inline]

6.470.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~~MessageMarshaller()` [inline, virtual]

6.470.3 Member Function Documentation

6.470.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 182), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 218), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 327), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 391), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 484), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 583).

6.470.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 183), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 328), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 485), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 584).

```
6.470.3.3 virtual int ac-
            tivemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1
            (OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs) throw ( decaf::io::IOException
            ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 183), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 328), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 485), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 584).

6.470.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 668).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 183), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 219), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 328), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 485), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 584).

6.470.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 669).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 220), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 329), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 352), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 486), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 585).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h`

6.471 cms::MessageNotReadableException Class Reference

This exception must be thrown when a CMS client attempts to read a write-only message.

#include <src/main/cms/MessageNotReadableException.h> Inheritance diagram for cms::MessageNotReadableException:

Public Member Functions

- **MessageNotReadableException** () throw ()
- **MessageNotReadableException** (const **MessageNotReadableException** &ex) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception *cause) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotReadableException** () throw ()

6.471.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

Since:

1.3

6.471.2 Constructor & Destructor Documentation

- 6.471.2.1 **cms::MessageNotReadableException::MessageNotReadableException** () throw ()
- 6.471.2.2 **cms::MessageNotReadableException::MessageNotReadableException** (const **MessageNotReadableException** & *ex*) throw ()
- 6.471.2.3 **cms::MessageNotReadableException::MessageNotReadableException** (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.471.2.4 **cms::MessageNotReadableException::MessageNotReadableException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.471.2.5 **virtual cms::MessageNotReadableException::~~MessageNotReadableException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageNotReadableException.h

6.472 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

#include <src/main/cms/MessageNotWriteableException.h> Inheritance diagram for cms::MessageNotWriteableException:

Public Member Functions

- **MessageNotWriteableException** () throw ()
- **MessageNotWriteableException** (const **MessageNotWriteableException** &ex) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception *cause) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotWriteableException** () throw ()

6.472.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since:

1.3

6.472.2 Constructor & Destructor Documentation

- 6.472.2.1 **cms::MessageNotWriteableException::MessageNotWriteableException** () throw ()
- 6.472.2.2 **cms::MessageNotWriteableException::MessageNotWriteableException** (const **MessageNotWriteableException** & *ex*) throw ()
- 6.472.2.3 **cms::MessageNotWriteableException::MessageNotWriteableException** (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.472.2.4 **cms::MessageNotWriteableException::MessageNotWriteableException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.472.2.5 virtual **cms::MessageNotWriteableException::~~MessageNotWriteableException** () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageNotWriteableException.h

6.473 cms::MessageProducer Class Reference

A client uses a `MessageProducer` (p. 2332) object to send messages to a **Destination** (p. 1480).

#include <src/main/cms/MessageProducer.h> Inheritance diagram for cms::MessageProducer:

Public Member Functions

- virtual `~MessageProducer ()`
- virtual void **send** (**Message** *message)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (**Message** *message, int deliveryMode, int priority, long long timeToLive)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **Destination** *destination, **Message** *message)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **Destination** *destination, **Message** *message, int deliveryMode, int priority, long long timeToLive)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **setDeliveryMode** (int mode)=0 throw (CMSEException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const =0 throw (CMSEException)
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value)=0 throw (CMSEException)
*Sets if **Message** (p. 2163) Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const =0 throw (CMSEException)
*Gets if **Message** (p. 2163) Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw (CMSEException)
*Sets if **Message** (p. 2163) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw (CMSEException)
*Gets if **Message** (p. 2163) Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority)=0 throw (CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const =0 throw (CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time)=0 throw (CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const =0 throw (CMSEException)
Gets the Time to Live that this producer sends messages with.

6.473.1 Detailed Description

A client uses a **MessageProducer** (p. 2332) object to send messages to a **Destination** (p. 1480). A **MessageProducer** (p. 2332) object is created by passing a **Destination** (p. 1480) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1480) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

Since:

1.0

6.473.2 Constructor & Destructor Documentation

6.473.2.1 virtual cms::MessageProducer::~MessageProducer () [inline, virtual]

6.473.3 Member Function Documentation

6.473.3.1 virtual int cms::MessageProducer::getDeliveryMode () const throw (CMSEException) [pure virtual]

Gets the delivery mode for this Producer.

Returns:

The **DeliveryMode** (p. 1478)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 958), and `activemq::core::ActiveMQProducer` (p. 408).

6.473.3.2 `virtual bool cms::MessageProducer::getDisableMessageID () const throw (CMSEException)` [pure virtual]

Gets if `Message` (p. 2163) Ids are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 959), and `activemq::core::ActiveMQProducer` (p. 408).

6.473.3.3 `virtual bool cms::MessageProducer::getDisableMessageTimeStamp () const throw (CMSEException)` [pure virtual]

Gets if `Message` (p. 2163) Time Stamps are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 959), and `activemq::core::ActiveMQProducer` (p. 408).

6.473.3.4 `virtual int cms::MessageProducer::getPriority () const throw (CMSEException)` [pure virtual]

Gets the Priority level that this producer sends messages at.

Returns:

int based priority level

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 959), and `activemq::core::ActiveMQProducer` (p. 409).

6.473.3.5 virtual long long cms::MessageProducer::getTimeToLive () const throw (CMSEException) [pure virtual]

Gets the Time to Live that this producer sends messages with.

Returns:

Time to live value in milliseconds

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 959), and `activemq::core::ActiveMQProducer` (p. 409).

6.473.3.6 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs while sending the message.

MessageFormatException (p. 2278) - if an Invalid **Message** (p. 2163) is given.

InvalidDestinationException (p. 1809) - if a client uses this method with a **MessageProducer** (p. 2332) with an invalid destination.

UnsupportedOperationException (p. 3306) - if a client uses this method with a **MessageProducer** (p. 2332) that did not specify a destination at creation time.

Implemented in `activemq::cmsutil::CachedProducer` (p. 960), and `activemq::core::ActiveMQProducer` (p. 410).

6.473.3.7 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*) throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

destination The destination on which to send the message

message the message to be sent.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs while sending the message.

MessageFormatException (p. 2278) - if an Invalid **Message** (p. 2163) is given.

InvalidDestinationException (p. 1809) - if a client uses this method with a **MessageProducer** (p. 2332) with an invalid destination.

UnsupportedOperationException (p. 3306) - if a client uses this method with a **MessageProducer** (p. 2332) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 960), and **activemq::core::ActiveMQProducer** (p. 410).

```
6.473.3.8 virtual void cms::MessageProducer::send (Message * message,
int deliveryMode, int priority, long long timeToLive)
throw ( cms::CMSEException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [pure virtual]
```

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs while sending the message.

MessageFormatException (p. 2278) - if an Invalid **Message** (p. 2163) is given.

InvalidDestinationException (p. 1809) - if a client uses this method with a **MessageProducer** (p. 2332) with an invalid destination.

UnsupportedOperationException (p. 3306) - if a client uses this method with a **MessageProducer** (p. 2332) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 961), and **activemq::core::ActiveMQProducer** (p. 411).

```
6.473.3.9 virtual void cms::MessageProducer::send (Message * message)
throw ( cms::CMSEException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [pure virtual]
```

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

message The message to be sent.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs while sending the message.

MessageFormatException (p. 2278) - if an Invalid **Message** (p. 2163) is given.

InvalidDestinationException (p. 1809) - if a client uses this method with a **MessageProducer** (p. 2332) with an invalid destination.

UnsupportedOperationException (p. 3306) - if a client uses this method with a **MessageProducer** (p. 2332) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 961), and **activemq::core::ActiveMQProducer** (p. 411).

6.473.3.10 virtual void cms::MessageProducer::setDeliveryMode (int *mode*) throw (CMSEException) [pure virtual]

Sets the delivery mode for this Producer.

Parameters:

mode The **DeliveryMode** (p. 1478)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 962), and **activemq::core::ActiveMQProducer** (p. 412).

6.473.3.11 virtual void cms::MessageProducer::setDisableMessageID (bool *value*) throw (CMSEException) [pure virtual]

Sets if **Message** (p. 2163) Ids are disabled for this Producer.

Parameters:

value boolean indicating enable / disable (true / false)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 962), and **activemq::core::ActiveMQProducer** (p. 412).

6.473.3.12 virtual void cms::MessageProducer::setDisableMessageTimeStamp (bool *value*) throw (CMSEException) [pure virtual]

Sets if **Message** (p. 2163) Time Stamps are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 962), and `activemq::core::ActiveMQProducer` (p. 412).

6.473.3.13 `virtual void cms::MessageProducer::setPriority (int priority) throw (CMSEException)` [pure virtual]

Sets the Priority that this Producers sends messages at.

Parameters:

priority int value for Priority level

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 962), and `activemq::core::ActiveMQProducer` (p. 412).

6.473.3.14 `virtual void cms::MessageProducer::setTimeToLive (long long time) throw (CMSEException)` [pure virtual]

Sets the Time to Live that this Producers sends messages with. This value will be used if the time to live is not specified via the send method.

Parameters:

time default time to live value in milliseconds

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 963), and `activemq::core::ActiveMQProducer` (p. 413).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageProducer.h`

6.474 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

```
#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
```

Public Member Functions

- **MessagePropertyInterceptor** (**commands::Message** *message, **util::PrimitiveMap** *properties) throw (**decaf::lang::exceptions::NullPointerException**)

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

- virtual **~MessagePropertyInterceptor** ()
- virtual bool **getBooleanProperty** (const std::string &name) const
Gets a boolean property.
- virtual unsigned char **getByteProperty** (const std::string &name) const
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const
Gets a float property.
- virtual int **getIntProperty** (const std::string &name) const
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)
Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)
Sets a string property.

6.474.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties. Currently the only properties that are intercepted and handled are:

Name		Conversion Supported	-----	JMSXDeliveryCount	
Int, Long, String		JMSXGroupID		String	
		JMSXGroupSeq		Int, Long, String	

6.474.2 Constructor & Destructor Documentation

6.474.2.1 activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInter (commands::Message * *message*, util::PrimitiveMap * *properties*) throw (decaf::lang::exceptions::NullPointerException)

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

Parameters:

message - The Message to store reserved property data in

properties - The PrimitiveMap to store the rest of the properties in.

Exceptions:

NullPointerException if either param is NULL

6.474.2.2 virtual
activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~~MessagePropertyInt
() [virtual]

6.474.3 Member Function Documentation

6.474.3.1 virtual bool ac-
tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty
(const std::string & *name*) const [virtual]

Gets a boolean property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.2 virtual unsigned char ac-
tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBytesProperty
(const std::string & *name*) const [virtual]

Gets a byte property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.3 virtual double ac-
tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty
(const std::string & *name*) const [virtual]

Gets a double property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.4 virtual float ac-
tivemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty
(const std::string & *name*) const [virtual]

Gets a float property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.5 `virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty(const std::string & name) const [virtual]`

Gets a int property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.6 `virtual long long activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty(const std::string & name) const [virtual]`

Gets a long property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.7 `virtual short activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty(const std::string & name) const [virtual]`

Gets a short property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.8 virtual std::string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty (const std::string & *name*) const [virtual]

Gets a string property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

6.474.3.9 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty (const std::string & *name*, bool *value*) [virtual]

Sets a boolean property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.10 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty (const std::string & *name*, unsigned char *value*) [virtual]

Sets a byte property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.11 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty (const std::string & *name*, double *value*) [virtual]

Sets a double property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.12 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty(const std::string & name, float value) [virtual]`

Sets a float property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.13 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty(const std::string & name, int value) [virtual]`

Sets a int property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.14 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty(const std::string & name, long long value) [virtual]`

Sets a long property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.15 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty(const std::string & name, short value) [virtual]`

Sets a short property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

6.474.3.16 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty(const std::string & name, const std::string & value) [virtual]`

Sets a string property.

Parameters:

- name* The name of the property to retrieve.
- value* The value for the named property.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h`

6.475 activemq::commands::MessagePull Class Reference

#include <src/main/activemq/commands/MessagePull.h> Inheritance diagram for activemq::commands::MessagePull:

Public Member Functions

- **MessagePull** ()
- virtual **~MessagePull** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessagePull * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &correlationId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEPULL** = 20

Protected Member Functions

- `MessagePull (const MessagePull &)`
- `MessagePull & operator= (const MessagePull &)`

Protected Attributes

- `Pointer< ConsumerId > consumerId`
- `Pointer< ActiveMQDestination > destination`
- `long long timeout`
- `std::string correlationId`
- `Pointer< MessageId > messageId`

6.475.1 Constructor & Destructor Documentation

6.475.1.1 `activemq::commands::MessagePull::MessagePull (const MessagePull &)`
[inline, protected]

6.475.1.2 `activemq::commands::MessagePull::MessagePull ()`

6.475.1.3 `virtual activemq::commands::MessagePull::~MessagePull ()` [virtual]

6.475.2 Member Function Documentation

6.475.2.1 `virtual MessagePull* activemq::commands::MessagePull::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.475.2.2 `virtual void activemq::commands::MessagePull::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.475.2.3 `virtual bool activemq::commands::MessagePull::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.475.2.4 `virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ()` [virtual]

6.475.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () const` [virtual]

6.475.2.6 `virtual std::string& activemq::commands::MessagePull::getCorrelationId ()` [virtual]

6.475.2.7 `virtual const std::string& activemq::commands::MessagePull::getCorrelationId () const` [virtual]

6.475.2.8 `virtual unsigned char activemq::commands::MessagePull::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.475.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination ()` [virtual]
- 6.475.2.10 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () const` [virtual]
- 6.475.2.11 `virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId ()` [virtual]
- 6.475.2.12 `virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () const` [virtual]
- 6.475.2.13 `virtual long long activemq::commands::MessagePull::getTimeout () const` [virtual]
- 6.475.2.14 `MessagePull& activemq::commands::MessagePull::operator= (const MessagePull &)` [inline, protected]
- 6.475.2.15 `virtual void activemq::commands::MessagePull::setConsumerId (const Pointer< ConsumerId > & consumerId)` [virtual]
- 6.475.2.16 `virtual void activemq::commands::MessagePull::setCorrelationId (const std::string & correlationId)` [virtual]
- 6.475.2.17 `virtual void activemq::commands::MessagePull::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.475.2.18 `virtual void activemq::commands::MessagePull::setMessageId (const Pointer< MessageId > & messageId)` [virtual]
- 6.475.2.19 `virtual void activemq::commands::MessagePull::setTimeout (long long timeout)` [virtual]
- 6.475.2.20 `virtual std::string activemq::commands::MessagePull::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p.655).

- 6.475.2.21 `virtual Pointer<Command> activemq::commands::MessagePull::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the

visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.475.3 Field Documentation

6.475.3.1 **Pointer<ConsumerId> activemq::commands::MessagePull::consumerId**
[protected]

6.475.3.2 **std::string activemq::commands::MessagePull::correlationId** [protected]

6.475.3.3 **Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination** [protected]

6.475.3.4 **const unsigned char activemq::commands::MessagePull::ID _-MESSAGEPULL = 20** [static]

6.475.3.5 **Pointer<MessageId> activemq::commands::MessagePull::messageId**
[protected]

6.475.3.6 **long long activemq::commands::MessagePull::timeout** [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

6.476 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2351).

#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.476.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2351). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.476.2 Constructor & Destructor Documentation

6.476.2.1 `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarshaller()` [inline]

6.476.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

6.476.3 Member Function Documentation

6.476.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.476.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.476.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.476.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.476.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.476.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.476.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

6.477 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2355).

#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.477.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2355). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.477.2 Constructor & Destructor Documentation

6.477.2.1 `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarshaller()` [inline]

6.477.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

6.477.3 Member Function Documentation

6.477.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.477.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.477.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.477.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.477.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.477.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.477.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h

6.478 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2359).

#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.478.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2359). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.478.2 Constructor & Destructor Documentation

6.478.2.1 `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarshaller()` [inline]

6.478.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

6.478.3 Member Function Documentation

6.478.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.478.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.478.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.478.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.478.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.478.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.478.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h

6.479 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2363).

#include <src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.479.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2363). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.479.2 Constructor & Destructor Documentation

6.479.2.1 `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::MessagePullMarshaller()` [inline]

6.479.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

6.479.3 Member Function Documentation

6.479.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.479.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.479.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.479.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.479.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.479.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.479.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h

6.480 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2367).

#include <src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.480.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p.2367). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.480.2 Constructor & Destructor Documentation

6.480.2.1 `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::MessagePullMarshaller()` [inline]

6.480.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::~~MessagePullMarshaller()` [inline, virtual]

6.480.3 Member Function Documentation

6.480.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.480.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.480.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.480.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.480.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.480.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.480.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

6.481 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 2371) defines a base level **Transport** (p. 3273) class that is intended to be used in place of an a regular protocol **Transport** (p. 3273) such as TCP.

#include <src/main/activemq/transport/mock/MockTransport.h> Inheritance diagram for activemq::transport::mock::MockTransport:

Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual ~**MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
*Sets the **ResponseBuilder** (p. 2785) that this class uses to create Responses to Commands sent.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setOutgoingListener** (**TransportListener** *listener)
Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat **AMQCPP_UNUSED**)
Sets the WireFormat instance to use.
- **Pointer**< **wireformat::WireFormat** > **getWireFormat** () const
Gets the currently set WireFormat.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **fireCommand** (const **Pointer**< **Command** > &command)

Fires a Command back through this transport to its registered CommandListener if there is one.

- virtual void **fireException** (const exceptions::ActiveMQException &ex)
Fires a Exception back through this transport to its registered ExceptionListener if there is one.
- virtual void **start** () throw (decaf::io::IOException)
*Starts the **Transport** (p. 3273), the send methods of a **Transport** (p. 3273) will throw an exception if used before the **Transport** (p. 3273) is started.*
- virtual void **stop** () throw (decaf::io::IOException)
*Stops the **Transport** (p. 3273).*
- virtual void **close** () throw (decaf::io::IOException)
Closes this object and deallocates the appropriate resources.
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3273) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3273) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException)
reconnect to another location
- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSentMessageBeforeFail** () const
- void **setNumSentMessageBeforeFail** (int value)
- int **getNumSentMessages** () const
- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const
- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const

- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)

Static Public Member Functions

- static **MockTransport** * **getInstance** ()

6.481.1 Detailed Description

The **MockTransport** (p. 2371) defines a base level **Transport** (p. 3273) class that is intended to be used in place of an a regular protocol **Transport** (p. 3273) such as TCP. This **Transport** (p. 3273) assumes that it is the base **Transport** (p. 3273) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 3273) defines an Interface **ResponseBuilder** (p. 2785) which must be implemented by any protocol for which the **Transport** (p. 3273) is used to Emulate. The **Transport** (p. 3273) hands off all outbound commands to the **ResponseBuilder** (p. 2785) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

6.481.2 Constructor & Destructor Documentation

6.481.2.1 **activemq::transport::mock::MockTransport::MockTransport** (const **Pointer**< **wireformat::WireFormat** > & *wireFormat*, const **Pointer**< **ResponseBuilder** > & *responseBuilder*)

6.481.2.2 **virtual activemq::transport::mock::MockTransport::~MockTransport** ()
[inline, virtual]

6.481.3 Member Function Documentation

6.481.3.1 **virtual void activemq::transport::mock::MockTransport::close** () **throw** (**decaf::io::IOException**) [virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

IOException if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 1019).

6.481.3.2 `virtual void activemq::transport::mock::MockTransport::fireCommand
(const Pointer< Command > & command) [inline, virtual]`

Fires a Command back through this transport to its registered CommandListener if there is one.

Parameters:

command - Command to send to the Listener.

6.481.3.3 `virtual void activemq::transport::mock::MockTransport::fireException
(const exceptions::ActiveMQException & ex) [inline, virtual]`

Fires a Exception back through this transport to its registered ExceptionListener if there is one.

Parameters:

ex The Exception that will be passed on the the **Transport** (p. 3273) listener.

6.481.3.4 `static MockTransport* ac-
tivemq::transport::mock::MockTransport::getInstance ()
[inline, static]`

6.481.3.5 `int ac-
tivemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail
() const [inline]`

6.481.3.6 `int ac-
tivemq::transport::mock::MockTransport::getNumReceivedMessages ()
const [inline]`

6.481.3.7 `int activemq::transport::mock::MockTransport::getNumSentKeepAlives
() const [inline]`

6.481.3.8 `int ac-
tivemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail
() const [inline]`

6.481.3.9 `int ac-
tivemq::transport::mock::MockTransport::getNumSentMessageBeforeFail
() const [inline]`

6.481.3.10 `int activemq::transport::mock::MockTransport::getNumSentMessages ()
const [inline]`

6.481.3.11 `virtual std::string ac-
tivemq::transport::mock::MockTransport::getRemoteAddress () const
[inline, virtual]`

Returns:

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3274).

6.481.3.12 `virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns:

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3274).

6.481.3.13 `Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat () const [inline]`

Gets the currently set WireFormat.

Returns:

the current WireFormat object.

6.481.3.14 `virtual bool activemq::transport::mock::MockTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3273) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 3273)

Implements **activemq::transport::Transport** (p. 3275).

6.481.3.15 `virtual bool activemq::transport::mock::MockTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3273) Connected to its Broker.

Returns:

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3275).

- 6.481.3.16 `bool activemq::transport::mock::MockTransport::isFailOnClose () const [inline]`
- 6.481.3.17 `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends () const [inline]`
- 6.481.3.18 `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage () const [inline]`
- 6.481.3.19 `bool activemq::transport::mock::MockTransport::isFailOnSendMessage () const [inline]`
- 6.481.3.20 `bool activemq::transport::mock::MockTransport::isFailOnStart () const [inline]`
- 6.481.3.21 `bool activemq::transport::mock::MockTransport::isFailOnStop () const [inline]`
- 6.481.3.22 `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 3273) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3275).

- 6.481.3.23 `virtual Transport* activemq::transport::mock::MockTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The `type_info` of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3275).

- 6.481.3.24 `virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3276).

6.481.3.25 `virtual void activemq::transport::mock::MockTransport::reconnect
(const decaf::net::URI &uri AMQCPP_UNUSED) throw (
decaf::io::IOException) [inline, virtual]`

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

6.481.3.26 `virtual Pointer<Response> ac-
tivemq::transport::mock::MockTransport::request
(const Pointer< Command > & command, un-
signed int timeout) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters:

command - The command to be sent.

timeout - The time to wait for this response.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3276).

6.481.3.27 `virtual Pointer<Response> ac-
tivemq::transport::mock::MockTransport::request (const
Pointer< Command > & command) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters:

command the command to be sent.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3277).

- 6.481.3.28 void activemq::transport::mock::MockTransport::setFailOnClose (bool *value*) [inline]
- 6.481.3.29 void activemq::transport::mock::MockTransport::setFailOnKeepAliveSends (bool *value*) [inline]
- 6.481.3.30 void activemq::transport::mock::MockTransport::setFailOnReceiveMessage (bool *value*) [inline]
- 6.481.3.31 void activemq::transport::mock::MockTransport::setFailOnSendMessage (bool *value*) [inline]
- 6.481.3.32 void activemq::transport::mock::MockTransport::setFailOnStart (bool *value*) [inline]
- 6.481.3.33 void activemq::transport::mock::MockTransport::setFailOnStop (bool *value*) [inline]
- 6.481.3.34 void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail (int *value*) [inline]
- 6.481.3.35 void activemq::transport::mock::MockTransport::setNumReceivedMessages (int *value*) [inline]
- 6.481.3.36 void activemq::transport::mock::MockTransport::setNumSentKeepAlives (int *value*) [inline]
- 6.481.3.37 void activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail (int *value*) [inline]
- 6.481.3.38 void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail (int *value*) [inline]
- 6.481.3.39 void activemq::transport::mock::MockTransport::setNumSentMessages (int *value*) [inline]
- 6.481.3.40 virtual void activemq::transport::mock::MockTransport::setOutgoingListener (TransportListener * *listener*) [inline, virtual]

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.

Parameters:

listener - The CommandListener to notify for each message

6.481.3.41 `void activemq::transport::mock::MockTransport::setResponseBuilder
(const Pointer< ResponseBuilder > & responseBuilder) [inline]`

Sets the **ResponseBuilder** (p. 2785) that this class uses to create Responses to Commands sent. These are either real Response Objects, or Commands that would have been sent Asynchronously be the Broker.

Parameters:

responseBuilder - The **ResponseBuilder** (p. 2785) to use from now on.

6.481.3.42 `virtual void activemq::transport::mock::MockTransport::setTransportListener
(TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters:

listener the listener of transport events.

Implements **activemq::transport::Transport** (p. 3277).

6.481.3.43 `virtual void activemq::transport::mock::MockTransport::setWireFormat
(const Pointer< wireformat::WireFormat > &wireFormat
AMQCPP_UNUSED) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters:

wireFormat WireFormat the object used to encode / decode commands.

6.481.3.44 `virtual void activemq::transport::mock::MockTransport::start () throw (
decaf::io::IOException) [virtual]`

Starts the **Transport** (p. 3273), the send methods of a **Transport** (p. 3273) will throw an exception if used before the **Transport** (p. 3273) is started.

Exceptions:

IOException if and error occurs while starting the **Transport** (p. 3273).

Implements **activemq::transport::Transport** (p. 3278).

6.481.3.45 `virtual void activemq::transport::mock::MockTransport::stop () throw (
decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3273).

Exceptions:

IOException if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 3278).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransport.h`

6.482 activemq::transport::mock::MockTransportFactory Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

#include <src/main/activemq/transport/mock/MockTransportFactory.h> Inheritance diagram for activemq::transport::mock::MockTransportFactory:

Public Member Functions

- virtual `~MockTransportFactory ()`
- virtual `Pointer< Transport > create (const decaf::net::URI &location) throw (exceptions::ActiveMQException)`

*Creates a fully configured **Transport** (p. 3273) instance which could be a chain of filters and transports.*

- virtual `Pointer< Transport > createComposite (const decaf::net::URI &location) throw (exceptions::ActiveMQException)`

*Creates a slimmed down **Transport** (p. 3273) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw (exceptions::ActiveMQException)`

*Creates a slimmed down **Transport** (p. 3273) instance which can be used in composite transport instances.*

6.482.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

6.482.2 Constructor & Destructor Documentation

6.482.2.1 virtual
activemq::transport::mock::MockTransportFactory::~MockTransportFactory
() [inline, virtual]

6.482.3 Member Function Documentation

6.482.3.1 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::create (const
decaf::net::URI & *location*) throw (exceptions::ActiveMQException)
[virtual]

Creates a fully configured **Transport** (p. 3273) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3279).

6.482.3.2 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::createComposite (const
decaf::net::URI & *location*) throw (exceptions::ActiveMQException)
[virtual]

Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3280).

6.482.3.3 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::doCreateComposite
(const decaf::net::URI & *location*, const Pointer<
wireformat::WireFormat > & *wireFormat*, const decaf::util::Properties
& *properties*) throw (exceptions::ActiveMQException) [protected,
virtual]

Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 3273).

properties - Properties to apply to the transport.

Returns:

Pointer to a new **Transport** (p. 3273) instance.

Exceptions:

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransportFactory.h`

6.483 decaf::util::concurrent::Mutex Class Reference

Mutex (p.2385) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

#include <src/main/decaf/util/concurrent/Mutex.h> Inheritance diagram for decaf::util::concurrent::Mutex:

Public Member Functions

- **Mutex** ()
- virtual ~**Mutex** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p.2023) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.483.1 Detailed Description

Mutex (p.2385) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

Since:

1.0

6.483.2 Constructor & Destructor Documentation

6.483.2.1 `decaf::util::concurrent::Mutex::Mutex ()`

6.483.2.2 `virtual decaf::util::concurrent::Mutex::~~Mutex () [virtual]`

6.483.3 Member Function Documentation

6.483.3.1 `virtual void decaf::util::concurrent::Mutex::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p.3123).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock()`, `decaf::util::StlMap< std::string, cms::Topic * >::lock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::lock()`.

6.483.3.2 `virtual void decaf::util::concurrent::Mutex::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p.3122) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p.3124).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify()`, `decaf::util::StlMap< std::string, cms::Topic * >::notify()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notify()`.

6.483.3.3 `virtual void decaf::util::concurrent::Mutex::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll()`, `decaf::util::StlMap< std::string, cms::Topic * >::notifyAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notifyAll()`.

6.483.3.4 `virtual bool decaf::util::concurrent::Mutex::tryLock () throw (
decaf::lang::exceptions::RuntimeException) [virtual]`

Attempts to **Lock** (p.2023) the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::tryLock()`, `decaf::util::StlMap< std::string, cms::Topic * >::tryLock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::tryLock()`.

6.483.3.5 `virtual void decaf::util::concurrent::Mutex::unlock () throw (
decaf::lang::exceptions::RuntimeException) [virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock()`, `decaf::util::StlMap< std::string, cms::Topic * >::unlock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::unlock()`.

6.483.3.6 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or `WAIT_INFINITE`

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.483.3.7 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or `WAIT_INFINITE`

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.483.3.8 **virtual void decaf::util::concurrent::Mutex::wait ()**
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait()`, `decaf::util::StlMap< std::string, cms::Topic * >::wait()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Mutex.h`

6.484 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/MutexHandle.h>
```

Public Member Functions

- **MutexHandle** ()
- **~MutexHandle** ()
- **MutexHandle** ()
- **~MutexHandle** ()

Data Fields

- pthread_mutex_t **mutex**
- volatile long long **lock_owner**
- volatile long long **lock_count**
- CRITICAL_SECTION **mutex**

6.484.1 Constructor & Destructor Documentation

6.484.1.1 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.484.1.2 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.484.1.3 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.484.1.4 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.484.2 Field Documentation

6.484.2.1 volatile long long decaf::util::concurrent::MutexHandle::lock_count

6.484.2.2 volatile long long decaf::util::concurrent::MutexHandle::lock_owner

6.484.2.3 CRITICAL_SECTION decaf::util::concurrent::MutexHandle::mutex

6.484.2.4 pthread_mutex_t decaf::util::concurrent::MutexHandle::mutex

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/**MutexHandle.h**
- src/main/decaf/internal/util/concurrent/windows/**MutexHandle.h**

6.485 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/MutexImpl.h>
```

Static Public Member Functions

- static **decaf::util::concurrent::MutexHandle * create** ()
Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.
- static void **destroy** (decaf::util::concurrent::MutexHandle *handle)
Destroy a previously create Mutex instance.
- static void **lock** (decaf::util::concurrent::MutexHandle *handle)
Locks the Mutex.
- static bool **trylock** (decaf::util::concurrent::MutexHandle *handle)
Tries to lock the Mutex.
- static void **unlock** (decaf::util::concurrent::MutexHandle *handle)
Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

6.485.1 Member Function Documentation

6.485.1.1 static decaf::util::concurrent::MutexHandle* decaf::internal::util::concurrent::MutexImpl::create () [static]

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

Returns:

handle to a newly created Mutex.

6.485.1.2 static void decaf::internal::util::concurrent::MutexImpl::destroy (decaf::util::concurrent::MutexHandle * handle) [static]

Destroy a previously create Mutex instance.

Parameters:

mutex The Mutex instance to be destroyed.

6.485.1.3 `static void decaf::internal::util::concurrent::MutexImpl::lock`
(`decaf::util::concurrent::MutexHandle * handle`) [static]

Locks the Mutex. If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

Parameters:

handle the handle to the Mutex to Lock.

6.485.1.4 `static bool decaf::internal::util::concurrent::MutexImpl::trylock`
(`decaf::util::concurrent::MutexHandle * handle`) [static]

Tries to lock the Mutex. If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

Parameters:

handle the handle to the Mutex to attempt to Lock.

Returns:

true if the lock was acquired false otherwise.

6.485.1.5 `static void decaf::internal::util::concurrent::MutexImpl::unlock`
(`decaf::util::concurrent::MutexHandle * handle`) [static]

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

Parameters:

handle the handle to the Mutex to attempt to Lock.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/MutexImpl.h`

6.486 activemq::commands::NetworkBridgeFilter Class Reference

#include <src/main/activemq/commands/NetworkBridgeFilter.h> Inheritance diagram for activemq::commands::NetworkBridgeFilter:

Public Member Functions

- **NetworkBridgeFilter** ()
- virtual **~NetworkBridgeFilter** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **NetworkBridgeFilter * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int networkTTL)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &networkBrokerId)

Static Public Attributes

- static const unsigned char **ID_NETWORKBRIDGEFILTER** = 91

Protected Member Functions

- **NetworkBridgeFilter** (const **NetworkBridgeFilter** &)
- **NetworkBridgeFilter** & **operator=** (const **NetworkBridgeFilter** &)

Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

6.486.1 Constructor & Destructor Documentation

6.486.1.1 `activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter (const NetworkBridgeFilter &) [inline, protected]`

6.486.1.2 `activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter ()`

6.486.1.3 `virtual
activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter ()
[virtual]`

6.486.2 Member Function Documentation

6.486.2.1 `virtual NetworkBridgeFilter* ac-
tivemq::commands::NetworkBridgeFilter::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.486.2.2 `virtual void ac-
tivemq::commands::NetworkBridgeFilter::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

6.486.2.3 `virtual bool activemq::commands::NetworkBridgeFilter::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

6.486.2.4 virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

6.486.2.5 virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () [virtual]

6.486.2.6 virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () const [virtual]

6.486.2.7 virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL () const [virtual]

6.486.2.8 NetworkBridgeFilter& activemq::commands::NetworkBridgeFilter::operator= (const NetworkBridgeFilter &) [inline, protected]

6.486.2.9 virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & *networkBrokerId*) [virtual]

6.486.2.10 virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int *networkTTL*) [virtual]

6.486.2.11 virtual std::string activemq::commands::NetworkBridgeFilter::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 718).

6.486.3 Field Documentation

6.486.3.1 `const unsigned char activemq::commands::NetworkBridgeFilter::ID_NETWORKBRIDGEFILTER = 91` [static]

6.486.3.2 `Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId` [protected]

6.486.3.3 `int activemq::commands::NetworkBridgeFilter::networkTTL` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/NetworkBridgeFilter.h`

6.487 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2397).

#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.487.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2397).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.487.2 Constructor & Destructor Documentation

6.487.2.1 `activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

6.487.2.2 `virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

6.487.3 Member Function Documentation

6.487.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.487.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.487.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.487.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.487.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.487.3.6 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.487.3.7 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h

6.488 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2401).

#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.488.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2401).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.488.2 Constructor & Destructor Documentation

6.488.2.1 `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

6.488.2.2 `virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

6.488.3 Member Function Documentation

6.488.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.488.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.488.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.488.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.488.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.488.3.6 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.488.3.7 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h

6.489 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2405).

#include <src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.489.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2405).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.489.2 Constructor & Destructor Documentation

6.489.2.1 `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

6.489.2.2 `virtual activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

6.489.3 Member Function Documentation

6.489.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.489.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.489.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.489.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseUnmarsh
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.489.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.489.3.6 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.489.3.7 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h

6.490 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2409).

#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.490.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2409).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.490.2 Constructor & Destructor Documentation

6.490.2.1 `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

6.490.2.2 `virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

6.490.3 Member Function Documentation

6.490.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.490.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.490.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.490.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.490.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.490.3.6 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.490.3.7 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h

6.491 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2413).

#include <src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.491.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2413).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.491.2 Constructor & Destructor Documentation

6.491.2.1 `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

6.491.2.2 `virtual activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

6.491.3 Member Function Documentation

6.491.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.491.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.491.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.491.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.491.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.491.3.6 virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.491.3.7 virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h

6.492 decaf::net::NoRouteToHostException Class Reference

#include <src/main/decaf/net/NoRouteToHostException.h> Inheritance diagram for decaf::net::NoRouteToHostException:

Public Member Functions

- **NoRouteToHostException** () throw ()
Default Constructor.
- **NoRouteToHostException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoRouteToHostException** (const **NoRouteToHostException** &ex) throw ()
Copy Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoRouteToHostException** (const std::exception *cause) throw ()
Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoRouteToHostException** * **clone** () const
Clones this exception.
- virtual ~**NoRouteToHostException** () throw ()

6.492.1 Constructor & Destructor Documentation

6.492.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException () throw () [inline]

Default Constructor.

6.492.1.2 decaf::net::NoRouteToHostException::NoRouteToHostException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.492.1.3 `decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.492.1.4 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.492.1.5 `decaf::net::NoRouteToHostException::NoRouteToHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.492.1.6 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.492.1.7 **virtual**
decaf::net::NoRouteToHostException::~~NoRouteToHostException ()
throw () [inline, virtual]

6.492.2 Member Function Documentation

6.492.2.1 **virtual NoRouteToHostException* de-**
caf::net::NoRouteToHostException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2973).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**NoRouteToHostException.h**

6.493 decaf::security::NoSuchAlgorithmException Class Reference

#include <src/main/decaf/security/NoSuchAlgorithmException.h> Inheritance diagram for decaf::security::NoSuchAlgorithmException:

Public Member Functions

- **NoSuchAlgorithmException** () throw ()
Default Constructor.
- **NoSuchAlgorithmException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchAlgorithmException** (const **NoSuchAlgorithmException** &ex) throw ()
Copy Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchAlgorithmException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchAlgorithmException** * **clone** () const
Clones this exception.
- virtual ~**NoSuchAlgorithmException** () throw ()

6.493.1 Constructor & Destructor Documentation

6.493.1.1 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException () throw () [inline]

Default Constructor.

6.493.1.2 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.493.1.3 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const NoSuchAlgorithmException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.493.1.4 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.493.1.5 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.493.1.6 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.493.1.7 **virtual**
decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException
() throw () [inline, virtual]

6.493.2 **Member Function Documentation**

6.493.2.1 **virtual NoSuchAlgorithmException* de-**
caf::security::NoSuchAlgorithmException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1708).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchAlgorithmException.h`

6.494 decaf::lang::exceptions::NoSuchElementException Class Reference

#include <src/main/decaf/lang/exceptions/NoSuchElementException.h> Inheritance diagram for decaf::lang::exceptions::NoSuchElementException:

Public Member Functions

- **NoSuchElementException** () throw ()
Default Constructor.
- **NoSuchElementException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()
Copy Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchElementException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchElementException** * clone () const
Clones this exception.
- virtual ~**NoSuchElementException** () throw ()

6.494.1 Constructor & Destructor Documentation

6.494.1.1 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException () throw () [inline]

Default Constructor.

6.494.1.2 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.494.1.3 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const NoSuchElementException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.494.1.4 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const char * file, const int lineNumber, const std::exception * cause,
const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.494.1.5 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.494.1.6 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException
(const char * file, const int lineNumber, const char * msg, ...) throw ()
[inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.494.1.7 **virtual**
decaf::lang::exceptions::NoSuchElementException::~~NoSuchElementException
() throw () [inline, virtual]

6.494.2 Member Function Documentation

6.494.2.1 **virtual** NoSuchElementException* de-
caf::lang::exceptions::NoSuchElementException::clone ()
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NoSuchElementException.h**

6.495 decaf::security::NoSuchProviderException Class Reference

#include <src/main/decaf/security/NoSuchProviderException.h> Inheritance diagram for decaf::security::NoSuchProviderException:

Public Member Functions

- **NoSuchProviderException** () throw ()
Default Constructor.
- **NoSuchProviderException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchProviderException** (const **NoSuchProviderException** &ex) throw ()
Copy Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchProviderException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchProviderException** * clone () const
Clones this exception.
- virtual ~**NoSuchProviderException** () throw ()

6.495.1 Constructor & Destructor Documentation

6.495.1.1 decaf::security::NoSuchProviderException::NoSuchProviderException () throw () [inline]

Default Constructor.

6.495.1.2 decaf::security::NoSuchProviderException::NoSuchProviderException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.495.1.3 decaf::security::NoSuchProviderException::NoSuchProviderException (const NoSuchProviderException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.495.1.4 decaf::security::NoSuchProviderException::NoSuchProviderException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.495.1.5 decaf::security::NoSuchProviderException::NoSuchProviderException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.495.1.6 decaf::security::NoSuchProviderException::NoSuchProviderException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.495.1.7 **virtual**
decaf::security::NoSuchProviderException::~~NoSuchProviderException
() throw () [inline, virtual]

6.495.2 Member Function Documentation

6.495.2.1 **virtual NoSuchProviderException* de-**
caf::security::NoSuchProviderException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1708).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchProviderException.h`

6.496 decaf::lang::exceptions::NullPointerException Class Reference

#include <src/main/decaf/lang/exceptions/NullPointerException.h> Inheritance diagram for decaf::lang::exceptions::NullPointerException:

Public Member Functions

- **NullPointerException** () throw ()
Default Constructor.
- **NullPointerException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()
Copy Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NullPointerException** (const std::exception *cause) throw ()
Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NullPointerException** * clone () const
Clones this exception.
- virtual ~**NullPointerException** () throw ()

6.496.1 Constructor & Destructor Documentation

6.496.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException () throw () [inline]

Default Constructor.

6.496.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.496.1.3 `decaf::lang::exceptions::NullPointerException::NullPointerException` `(const NullPointerException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.496.1.4 `decaf::lang::exceptions::NullPointerException::NullPointerException` `(const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.496.1.5 `decaf::lang::exceptions::NullPointerException::NullPointerException` `(const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.496.1.6 `decaf::lang::exceptions::NullPointerException::NullPointerException` `(const char * file, const int lineNumber, const char * msg, ...) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.496.1.7 **virtual**
decaf::lang::exceptions::NullPointerException::~~NullPointerException ()
throw () [inline, virtual]

6.496.2 Member Function Documentation

6.496.2.1 **virtual NullPointerException* de-**
caf::lang::exceptions::NullPointerException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NullPointerException.h**

6.497 decaf::lang::Number Class Reference

The abstract class **Number** (p. 2432) is the superclass of classes **Byte** (p. 840), **Double** (p. 1537), **Float** (p. 1647), **Integer** (p. 1762), **Long** (p. 2057), and **Short** (p. 2906).

#include <src/main/decaf/lang/Number.h> Inheritance diagram for decaf::lang::Number:

Public Member Functions

- virtual **~Number** ()
• virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual double **doubleValue** () const =0
Answers the double value which the receiver represents.
- virtual float **floatValue** () const =0
Answers the float value which the receiver represents.
- virtual int **intValue** () const =0
Answers the int value which the receiver represents.
- virtual long long **longValue** () const =0
Answers the long value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.

6.497.1 Detailed Description

The abstract class **Number** (p. 2432) is the superclass of classes **Byte** (p. 840), **Double** (p. 1537), **Float** (p. 1647), **Integer** (p. 1762), **Long** (p. 2057), and **Short** (p. 2906). Subclasses of **Number** (p. 2432) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

6.497.2 Constructor & Destructor Documentation

6.497.2.1 virtual decaf::lang::Number::~~Number () [inline, virtual]

6.497.3 Member Function Documentation

6.497.3.1 virtual unsigned char decaf::lang::Number::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 842), **decaf::lang::Character** (p. 984), **decaf::lang::Double** (p. 1539), **decaf::lang::Float** (p. 1649), **decaf::lang::Integer** (p. 1765), **decaf::lang::Long** (p. 2060), and **decaf::lang::Short** (p. 2908).

6.497.3.2 virtual double decaf::lang::Number::doubleValue () const [pure virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 843), **decaf::lang::Character** (p. 985), **decaf::lang::Double** (p. 1541), **decaf::lang::Float** (p. 1650), **decaf::lang::Integer** (p. 1766), **decaf::lang::Long** (p. 2061), **decaf::lang::Short** (p. 2909), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 635).

6.497.3.3 virtual float decaf::lang::Number::floatValue () const [pure virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 844), **decaf::lang::Character** (p. 985), **decaf::lang::Double** (p. 1542), **decaf::lang::Float** (p. 1652), **decaf::lang::Integer** (p. 1767), **decaf::lang::Long** (p. 2062), **decaf::lang::Short** (p. 2910), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 635).

6.497.3.4 virtual int decaf::lang::Number::intValue () const [pure virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 844), **decaf::lang::Character** (p. 986), **decaf::lang::Double** (p. 1542), **decaf::lang::Float** (p. 1653), **decaf::lang::Integer** (p. 1767), **decaf::lang::Long** (p. 2062), **decaf::lang::Short** (p. 2910), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 637).

6.497.3.5 virtual long long decaf::lang::Number::longValue () const [pure virtual]

Answers the long value which the receiver represents.

Returns:

long long the value of the receiver.

Implemented in `decaf::lang::Byte` (p. 844), `decaf::lang::Character` (p. 987), `decaf::lang::Double` (p. 1543), `decaf::lang::Float` (p. 1653), `decaf::lang::Integer` (p. 1768), `decaf::lang::Long` (p. 2063), `decaf::lang::Short` (p. 2910), and `decaf::util::concurrent::atomic::AtomicInteger` (p. 637).

6.497.3.6 virtual short decaf::lang::Number::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented in `decaf::lang::Byte` (p. 846), `decaf::lang::Character` (p. 988), `decaf::lang::Double` (p. 1545), `decaf::lang::Float` (p. 1655), `decaf::lang::Integer` (p. 1772), `decaf::lang::Long` (p. 2067), and `decaf::lang::Short` (p. 2912).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Number.h`

6.498 decaf::lang::exceptions::NumberFormatException Class Reference

#include <src/main/decaf/lang/exceptions/NumberFormatException.h> Inheritance diagram for decaf::lang::exceptions::NumberFormatException:

Public Member Functions

- **NumberFormatException** ()
Default Constructor.
- **NumberFormatException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **NumberFormatException** (const **NumberFormatException** &ex) throw ()
Copy Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NumberFormatException** (const std::exception *cause) throw ()
Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NumberFormatException** * **clone** () const
Clones this exception.
- virtual ~**NumberFormatException** () throw ()

6.498.1 Constructor & Destructor Documentation

6.498.1.1 decaf::lang::exceptions::NumberFormatException::NumberFormatException () [inline]

Default Constructor.

Referenced by clone().

6.498.1.2 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.498.1.3 `decaf::lang::exceptions::NumberFormatException::NumberFormatException
(const NumberFormatException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.498.1.4 `decaf::lang::exceptions::NumberFormatException::NumberFormatException
(const char * file, const int lineNumber, const std::exception * cause,
const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

References `decaf::lang::Exception::buildMessage()`, and `decaf::lang::Exception::setMark()`.

6.498.1.5 `decaf::lang::exceptions::NumberFormatException::NumberFormatException
(const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.498.1.6 `decaf::lang::exceptions::NumberFormatException::NumberFormatException
(const char * file, const int lineNumber, const char * msg, ...) throw ()
[inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

6.498.1.7 virtual

decaf::lang::exceptions::NumberFormatException::~~NumberFormatException
() throw () [inline, virtual]

6.498.2 Member Function Documentation

6.498.2.1 virtual NumberFormatException* de-
caf::lang::exceptions::NumberFormatException::clone ()
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

References NumberFormatException().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NumberFormatException.h**

6.499 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

`#include <src/main/cms/ObjectMessage.h>`Inheritance diagram for cms::ObjectMessage:

Public Member Functions

- virtual `~ObjectMessage ()`

6.499.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object. serialized `ObjectMessage` (p. 2438)s.

Since:

1.0

6.499.2 Constructor & Destructor Documentation

6.499.2.1 virtual cms::ObjectMessage::~ObjectMessage () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/ObjectMessage.h`

6.500 decaf::security_provider::unix::openssl::OpenSSLX500Principal Class Reference

The `OpenSSLX500Principal` (p. 2439) wraps around an OpenSSL `X509_NAME` structure.

```
#include <src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h>
```

Public Member Functions

- **OpenSSLX500Principal** (`X509_NAME *name`)
Constructor.
- virtual **~OpenSSLX500Principal** ()
Destructor.
- virtual `X509_NAME *getX509Name` ()
Accessor to the underlying X509 name structure.
- virtual `bool equals` (`const Principal &another`) `const`
Compares two principals to see if they are the same.
- virtual `std::string getName` () `const`
Returns the distinguished name string using the RFC2253 formatting.
- virtual `void getEncoded` (`std::vector< unsigned char > &output`) `const`
Serializes the distinguished name to its ASN.1 DER encoded form.

Static Public Member Functions

- static `void getEncoded` (`X509_NAME *name`, `std::vector< unsigned char > &output`)
Serializes the given distinguished name to its ASN.1 DER encoded form.
- static `std::string toString` (`X509_NAME *name`) `const`
Converts the given name to a string using the RFC2253 formatting.

6.500.1 Detailed Description

The `OpenSSLX500Principal` (p. 2439) wraps around an OpenSSL `X509_NAME` structure. It does not, however, control the lifetime of the structure.

6.500.2 Constructor & Destructor Documentation

6.500.2.1 decaf::security_provider::unix::openssl::OpenSSLX500Principal::OpenSSLX500Principal (`X509_NAME * name`)

Constructor. Saves the internal X509 name and caches the string representation of the name.

Parameters:

name The underlying X509 name structure.

```
6.500.2.2  virtual decaf::security_-
            provider::unix::openssl::OpenSSLX500Principal::~~OpenSSLX500Principal
            () [inline, virtual]
```

Destructor. Does nothing.

6.500.3 Member Function Documentation

```
6.500.3.1  virtual bool decaf::security_-
            provider::unix::openssl::OpenSSLX500Principal::equals
            (const Principal & another) const [virtual]
```

Compares two principals to see if they are the same.

Parameters:

another A principal to be tested for equality to this one.

Returns:

true if the given principal is equivalent to this one.

```
6.500.3.2  static void decaf::security_-
            provider::unix::openssl::OpenSSLX500Principal::getEncoded
            (X509_NAME * name, std::vector< unsigned char > & output)
            [static]
```

Serializes the given distinguished name to its ASN.1 DER encoded form.

Parameters:

name the X509 name structure to be encoded.

output Receives the distinguished name in ASN.1 DER encoded form.

```
6.500.3.3  virtual void decaf::security_-
            provider::unix::openssl::OpenSSLX500Principal::getEncoded
            (std::vector< unsigned char > & output) const [inline, virtual]
```

Serializes the distinguished name to its ASN.1 DER encoded form.

Parameters:

output Receives the distinguished name in ASN.1 DER encoded form.

6.500.3.4 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX500Principal::getName()
() const [inline, virtual]

Returns the distinguished name string using the RFC2253 formatting.

Returns:

the RFC2253 formatted distinguished name string.

References toString().

6.500.3.5 virtual X509_NAME* decaf::security_provider::unix::openssl::OpenSSLX500Principal::getX509Name()
[inline, virtual]

Accessor to the underlying X509 name structure.

6.500.3.6 static std::string decaf::security_provider::unix::openssl::OpenSSLX500Principal::toString(X509_NAME * name) const
[static]

Converts the given name to a string using the RFC2253 formatting.

Parameters:

name the X509 name structure to be formatted.

Returns:

the RFC2253 formatted name string.

Referenced by getName().

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h

6.501 decaf::security_provider::unix::openssl::OpenSSLX509Certificate Class Reference

#include <src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h> Inheritance diagram for decaf::security_provider::unix::openssl::OpenSSLX509Certificate:

Public Member Functions

- virtual **~OpenSSLX509Certificate** ()
- virtual bool **equals** (const Certificate &cert) const =0
Compares the encoded form of the two certificates.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
Provides the encoded form of this certificate.
- virtual std::string **getType** () const =0
Returns the type of this certificate.
- virtual PublicKey * **getPublicKey** ()=0
Gets the public key of this certificate.
- virtual const PublicKey * **getPublicKey** () const =0
Gets the public key of this certificate.
- virtual void **verify** (const PublicKey &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual void **verify** (const PublicKey &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual std::string **toString** () const =0
Returns a string representation of this certificate.
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const decaf::util::Date &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0

- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
- virtual int **getVersion** () const =0

6.501.1 Constructor & Destructor Documentation

6.501.1.1 virtual decaf::security_provider::unix::openssl::OpenSSLX509Certificate::~~OpenSSLX509Certificate () [virtual]

6.501.2 Member Function Documentation

6.501.2.1 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3407).

6.501.2.2 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::checkValidity () const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3408).

6.501.2.3 virtual bool decaf::security_provider::unix::openssl::OpenSSLX509Certificate::equals (const Certificate & *cert*) const [pure virtual]

Compares the encoded form of the two certificates.

Parameters:

cert The certificate to be tested for equality with this certificate.

Returns:

true if the given certificate is equal to this certificate.

6.501.2.4 `virtual int decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getBasicConstraints ()
const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.5 `virtual void decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getEncoded
(std::vector< unsigned char > & output) const throw (
CertificateEncodingException) [pure virtual]`

Provides the encoded form of this certificate.

Parameters:

output Receives the encoded form of this certificate.

Exceptions:

CertificateEncodingException if an encoding error occurs

Implements `decaf::security::cert::Certificate` (p. 969).

6.501.2.6 `virtual void decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getIssuerUniqueID
(std::vector< bool > & output) const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.7 `virtual const X500Principal* decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getIssuerX500Principal
() const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.8 `virtual void decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getKeyUsage
(std::vector< unsigned char > & output) const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.9 `virtual Date decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getNotAfter () const
[pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.10 `virtual Date decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getNotBefore () const
[pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.11 `virtual const PublicKey* decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getPublicKey () const
[pure virtual]`

Gets the public key of this certificate.

Returns:

the public key

Implements `decaf::security::cert::Certificate` (p. 969).

6.501.2.12 `virtual PublicKey* decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getPublicKey ()
[pure virtual]`

Gets the public key of this certificate.

Returns:

the public key

Implements `decaf::security::cert::Certificate` (p. 969).

6.501.2.13 `virtual std::string decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getSigAlgName ()
const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3408).

6.501.2.14 `virtual std::string decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getSigAlgOID ()
const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.15 `virtual void decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getSigAlgParams
(std::vector< unsigned char > & output) const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.16 `virtual void decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getSignature
(std::vector< unsigned char > & output) const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.17 `virtual void decaf::security_ -
provider::unix::openssl::OpenSSLX509Certificate::getSubjectUniqueID
(std::vector< bool > & output) const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.18 `virtual const X500Principal* decaf::security_ -
provider::unix::openssl::OpenSSLX509Certificate::getSubjectX500Principal
() const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.19 `virtual void decaf::security_ -
provider::unix::openssl::OpenSSLX509Certificate::getTBSCertificate
(std::vector< unsigned char > & output) const throw (
CertificateEncodingException) [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.20 `virtual std::string decaf::security_ -
provider::unix::openssl::OpenSSLX509Certificate::getType () const
[pure virtual]`

Returns the type of this certificate.

Returns:

the type of this certificate

Implements `decaf::security::cert::Certificate` (p. 970).

6.501.2.21 `virtual int decaf::security_ -
provider::unix::openssl::OpenSSLX509Certificate::getVersion () const
[pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3409).

6.501.2.22 `virtual std::string decaf::security_ -
provider::unix::openssl::OpenSSLX509Certificate::toString () const
[pure virtual]`

Returns a string representation of this certificate.

Returns:

a string representation of this certificate

Implements `decaf::security::cert::Certificate` (p. 970).

6.501.2.23 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::verify (const PublicKey & *publicKey*, const std::string & *sigProvider*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key. Uses the verification engine of the specified provider.

Parameters:

publicKey The public key used to carry out the validation.

sigProvider The name of the signature provider

Exceptions:

NoSuchAlgorithmException - on unsupported signature algorithms.

InvalidKeyException - on incorrect key.

NoSuchProviderException - if there's no default provider.

SignatureException - on signature errors.

CertificateException - on encoding errors.

6.501.2.24 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::verify (const PublicKey & *publicKey*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters:

publicKey The public key used to carry out the validation.

Exceptions:

NoSuchAlgorithmException - on unsupported signature algorithms.

InvalidKeyException - on incorrect key.

NoSuchProviderException - if there's no default provider.

SignatureException - on signature errors.

CertificateException - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h

6.502 activemq::wireformat::openwire::OpenWireFormat Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireFormat:

Public Member Functions

- **OpenWireFormat** (const **decaf::util::Properties** &properties)
*Constructs a new **OpenWireFormat** (p. 2448) object.*
- virtual **~OpenWireFormat** ()
- virtual bool **hasNegotiator** () const
*Returns true if this **WireFormat** (p. 3363) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual **Pointer< transport::Transport > createNegotiator** (const **Pointer< transport::Transport >** &transport) throw (**decaf::lang::exceptions::UnsupportedOperationException**)
If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.
- void **addMarshaller** (**marshal::DataStreamMarshaller** *marshaller)
Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.
- virtual void **marshal** (const **Pointer< commands::Command >** &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out) throw (**decaf::io::IOException**)
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- virtual **Pointer< commands::Command > unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in) throw (**decaf::io::IOException**)
Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.
- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** *object, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Utility method for Tight Marshaling the given object to the boolean stream passed.
- void **tightMarshalNestedObject2** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *ds, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

- **commands::DataStructure** * **tightUnmarshalNestedObject** (decaf::io::DataInputStream *dis, utils::BooleanStream *bs) throw (decaf::io::IOException)
Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.
- **commands::DataStructure** * **looseUnmarshalNestedObject** (decaf::io::DataInputStream *dis) throw (decaf::io::IOException)
Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.
- void **looseMarshalNestedObject** (commands::DataStructure *o, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Utility method to loosely Marshal an object that is derived from the DataStrcutre interface.
- void **renegotiateWireFormat** (const commands::WireFormatInfo &info) throw (decaf::lang::exceptions::IllegalStateException)
Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.
- virtual void **setPreferedWireFormatInfo** (const Pointer< commands::WireFormatInfo > &info) throw (decaf::lang::exceptions::IllegalStateException)
Configures this object using the provided WireformatInfo object.
- virtual const Pointer< commands::WireFormatInfo > & **getPreferedWireFormatInfo** () const
Gets the Preferred WireFormatInfo object that this class holds.
- bool **isStackTraceEnabled** () const
Checks if the stackTraceEnabled flag is on.
- void **setStackTraceEnabled** (bool stackTraceEnabled)
Sets if the stackTraceEnabled flag is on.
- bool **isTcpNoDelayEnabled** () const
Checks if the tcpNoDelayEnabled flag is on.
- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)
Sets if the tcpNoDelayEnabled flag is on.
- int **getVersion** () const
Get the current Wireformat Version.
- void **setVersion** (int version) throw (decaf::lang::exceptions::IllegalArgumentException)
Set the current Wireformat Version.
- virtual bool **inReceive** () const
Is there a Message being unmarshaled?
- bool **isCacheEnabled** () const

Checks if the cacheEnabled flag is on.

- void **setCacheEnabled** (bool cacheEnabled)
Sets if the cacheEnabled flag is on.
- int **getCacheSize** () const
Returns the currently set Cache size.
- void **setCacheSize** (int value)
Sets the current Cache size.
- bool **isTightEncodingEnabled** () const
Checks if the tightEncodingEnabled flag is on.
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)
Sets if the tightEncodingEnabled flag is on.
- bool **isSizePrefixDisabled** () const
Checks if the sizePrefixDisabled flag is on.
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)
Sets if the sizePrefixDisabled flag is on.
- long long **getMaxInactivityDuration** () const
Gets the MaxInactivityDuration setting.
- void **setMaxInactivityDuration** (long long value)
Sets the MaxInactivityDuration setting.
- long long **getMaxInactivityDurationInitialDelay** () const
Gets the MaxInactivityDurationInitialDelay setting.
- void **setMaxInactivityDurationInitialDelay** (long long value)
Sets the MaxInactivityDurationInitialDelay setting.

Protected Member Functions

- **commands::DataStructure * doUnmarshal** (decaf::io::DataInputStream *dis)
throw (decaf::io::IOException)
Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.
- void **destroyMarshallers** ()
Cleans up all registered Marshallers and empties the dataMarshallers vector.

Static Protected Attributes

- static const unsigned char **NULL_TYPE**
- static const int **DEFAULT_VERSION** = 1

6.502.1 Constructor & Destructor Documentation

6.502.1.1 activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat (const decaf::util::Properties & *properties*)

Constructs a new **OpenWireFormat** (p. 2448) object.

Parameters:

properties - can contain optional config params.

6.502.1.2 virtual activemq::wireformat::openwire::OpenWireFormat::~~OpenWireFormat () [virtual]

6.502.2 Member Function Documentation

6.502.2.1 void activemq::wireformat::openwire::OpenWireFormat::addMarshaller (marshal::DataStreamMarshaller * *marshaller*)

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

Parameters:

marshaller - the Marshaler to add to the collection.

6.502.2.2 virtual Pointer<transport::Transport> activemq::wireformat::openwire::OpenWireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns:

new instance of a **WireFormatNegotiator** (p. 3399).

Implements **activemq::wireformat::WireFormat** (p. 3364).

6.502.2.3 void activemq::wireformat::openwire::OpenWireFormat::destroyMarshallers () [protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector. This should be called before a reconfiguration of the version marshallers, or on destruction of this object

6.502.2.4 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::doUnmarshal (decaf::io::DataInputStream * dis) throw (decaf::io::IOException)`
[protected]

Perform the actual unmarshal of data from the given `DataInputStream` return the unmarshalled `DataStructure` object once done, caller takes ownership of this object. This method can return null if the type of the object to unmarshal is NULL, empty data.

Parameters:

dis - `DataInputStream` to read from

Returns:

new `DataStructure*` that the caller owns

Exceptions:

IOException if an error occurs during the unmarshal

6.502.2.5 `int activemq::wireformat::openwire::OpenWireFormat::getCacheSize ()`
`const` [inline]

Returns the currently set Cache size.

Returns:

the current value of the broker's cache size.

6.502.2.6 `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration ()`
`const` [inline]

Gets the `MaxInactivityDuration` setting.

Returns:

maximum inactivity duration value in milliseconds.

6.502.2.7 `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay ()`
`const` [inline]

Gets the `MaxInactivityDurationInitialDelay` setting.

Returns:

maximum inactivity duration initial delay value in milliseconds.

6.502.2.8 `virtual const Pointer<commands::WireFormatInfo>& activemq::wireformat::openwire::OpenWireFormat::getPreferredWireFormatInfo () const [inline, virtual]`

Gets the Preferred WireFormatInfo object that this class holds.

Returns:

pointer to a preferred WireFormatInfo object

6.502.2.9 `int activemq::wireformat::openwire::OpenWireFormat::getVersion () const [inline, virtual]`

Get the current Wireformat Version.

Returns:

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3364).

6.502.2.10 `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator () const [inline, virtual]`

Returns true if this **WireFormat** (p. 3363) has a Negotiator that needs to wrap the Transport that uses it.

Returns:

true if the **WireFormat** (p. 3363) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3364).

6.502.2.11 `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive () const [inline, virtual]`

Is there a Message being unmarshaled?

Returns:

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3365).

6.502.2.12 `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled () const [inline]`

Checks if the cacheEnabled flag is on.

Returns:

true if the flag is on.

6.502.2.13 `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled () const [inline]`

Checks if the sizePrefixDisabled flag is on.

Returns:

true if the flag is on.

6.502.2.14 `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled () const [inline]`

Checks if the stackTraceEnabled flag is on.

Returns:

true if the flag is on.

6.502.2.15 `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled () const [inline]`

Checks if the tcpNoDelayEnabled flag is on.

Returns:

true if the flag is on.

6.502.2.16 `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled () const [inline]`

Checks if the tightEncodingEnabled flag is on.

Returns:

true if the flag is on.

6.502.2.17 `void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (commands::DataStructure * o, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)`

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface. The marshaled data is written to the passed in DataOutputStream.

Parameters:

o - DataStructure derived Object to Marshal

dataOut - DataOutputStream to write the data to

Exceptions:

IOException if an error occurs.

6.502.2.18 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (decaf::io::DataInputStream * dis) throw (decaf::io::IOException)`

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format. Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

Parameters:

dis - the DataInputStream to read the data from

Returns:

a new DataStructure derived Object pointer

Exceptions:

IOException if an error occurs.

6.502.2.19 `virtual void activemq::wireformat::openwire::OpenWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw (decaf::io::IOException) [virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters:

command The Command to Marshal.

transport The Transport instance that called this method.

out The output stream to write the command to.

Exceptions:

IOException

Implements `activemq::wireformat::WireFormat` (p. 3365).

6.502.2.20 `void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat (const commands::WireFormatInfo & info) throw (decaf::lang::exceptions::IllegalStateException)`

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

Parameters:

info - The new Wireformat Info settings

Exceptions:

IllegalStateException is wire format can't be negotiated.

6.502.2.21 `void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled (bool cacheEnabled) [inline]`

Sets if the cacheEnabled flag is on.

Parameters:

cacheEnabled - true to turn flag is on

6.502.2.22 `void activemq::wireformat::openwire::OpenWireFormat::setCacheSize (int value) [inline]`

Sets the current Cache size.

Parameters:

value - the value to send as the broker's cache size.

6.502.2.23 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long value) [inline]`

Sets the MaxInactivityDuration setting.

Parameters:

value - the Max inactivity duration value in milliseconds.

6.502.2.24 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay (long long value) [inline]`

Sets the MaxInactivityDurationInitialDelay setting.

Parameters:

value - the Max inactivity Initial Delay duration value in milliseconds.

6.502.2.25 `virtual void activemq::wireformat::openwire::OpenWireFormat::setPreferredWireFormatInfo (const Pointer< commands::WireFormatInfo > & info) throw (decaf::lang::exceptions::IllegalStateException) [virtual]`

Configures this object using the provided WireformatInfo object.

Parameters:

info - a WireFormatInfo object, takes ownership.

6.502.2.26 void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled (bool *sizePrefixDisabled*) [inline]

Sets if the sizePrefixDisabled flag is on.

Parameters:

sizePrefixDisabled - true to turn flag is on

6.502.2.27 void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled (bool *stackTraceEnabled*) [inline]

Sets if the stackTraceEnabled flag is on.

Parameters:

stackTraceEnabled - true to turn flag is on

6.502.2.28 void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*) [inline]

Sets if the tcpNoDelayEnabled flag is on.

Parameters:

tcpNoDelayEnabled - true to turn flag is on

6.502.2.29 void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool *tightEncodingEnabled*) [inline]

Sets if the tightEncodingEnabled flag is on.

Parameters:

tightEncodingEnabled - true to turn flag is on

6.502.2.30 void activemq::wireformat::openwire::OpenWireFormat::setVersion (int *version*) throw (decaf::lang::exceptions::IllegalArgumentException) [virtual]

Set the current Wireformat Version.

Parameters:

version - int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3365).

6.502.2.31 `virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1(commands::DataStructure * object, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Utility method for Tight Marshaling the given object to the boolean stream passed.

Parameters:

object - The DataStructure to marshal

bs - the BooleanStream to write to

Returns:

size of the data returned.

6.502.2.32 `void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2(commands::DataStructure * o, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface. Writes the data to the Data Output Stream provided.

Parameters:

o - DataStructure object

ds - DataOutputStream for writing

bs - BooleanStream

Exceptions:

IOException if an error occurs.

6.502.2.33 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject(decaf::io::DataInputStream * dis, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream. The DataStructure instance that is returned is now the property of the caller.

Parameters:

dis - DataInputStream to read from

bs - BooleanStream to read from

Returns:

Newly allocated DataStructure Object

Exceptions:

IOException if an error occurs.

6.502.2.34 virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form. Returns a Pointer to the newly un-marshaled Command.

Parameters:

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns:

the newly marshaled Command, caller owns the pointer

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 3366).

6.502.3 Field Documentation

6.502.3.1 const int
activemq::wireformat::openwire::OpenWireFormat::DEFAULT_VERSION = 1 [static, protected]

6.502.3.2 const unsigned char
activemq::wireformat::openwire::OpenWireFormat::NULL_TYPE [static, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/OpenWireFormat.h

6.503 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatFactory:

Public Member Functions

- **OpenWireFormatFactory ()**

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

- **virtual ~OpenWireFormatFactory ()**

- **virtual Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties) throw (decaf::lang::exceptions::IllegalStateException)**

*Creates a new **WireFormat** (p. 3363) Object passing it a set of properties from which it can obtain any optional settings.*

6.503.1 Constructor & Destructor Documentation

6.503.1.1 activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory () [inline]

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called. URL options ----- wireFormat.stackTraceEnabled wireFormat.cacheEnabled wireFormat.tcpNoDelayEnabled wireFormat.tightEncodingEnabled wireFormat.sizePrefixDisabled wireFormat.maxInactivityDuration wireFormat.maxInactivityDurationInitialDelay

6.503.1.2 virtual activemq::wireformat::openwire::OpenWireFormatFactory::~~OpenWireFormatFactory () [inline, virtual]

6.503.2 Member Function Documentation

6.503.2.1 virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalStateException) [virtual]

Creates a new **WireFormat** (p. 3363) Object passing it a set of properties from which it can obtain any optional settings.

Parameters:

properties - the Properties for this **WireFormat** (p. 3363)

Implements **activemq::wireformat::WireFormatFactory** (p. 3367).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h`

6.504 activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatNegotiator:

Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** *wireFormat, const **Pointer**< **transport::Transport** > &next)

Constructor - Initializes this object around another Transport.

- virtual ~**OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends a one-way command.

- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends the given request to the server and waits for the response.

- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends the given request to the server and waits for the response.

- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)

This is called in the context of the nested transport's reading thread.

- virtual void **onTransportException** (**transport::Transport** *source, const **decaf::lang::Exception** &ex)

Event handler for an exception from a command transport.

- virtual void **start** () throw (decaf::io::IOException)

Starts this transport object and creates the thread for polling on the input stream for commands.

- virtual void **close** () throw (decaf::io::IOException)

Stops the polling thread and closes the streams.

6.504.1 Constructor & Destructor Documentation

6.504.1.1 activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator (**OpenWireFormat** * wireFormat, const **Pointer**< **transport::Transport** > & next)

Constructor - Initializes this object around another Transport.

Parameters:

wireFormat - The **WireFormat** (p. 3363) object we use to negotiate

next - The next transport in the chain

6.504.1.2 virtual
activemq::wireformat::openwire::OpenWireFormatNegotiator::~~OpenWireFormatNegotiator()
[virtual]

6.504.2 Member Function Documentation

6.504.2.1 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::close ()
throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

IOException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3283).

6.504.2.2 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand
(const Pointer< commands::Command > & *command*) [virtual]

This is called in the context of the nested transport's reading thread. In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters:

command the received from the nested transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3285).

6.504.2.3 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway
(const Pointer< commands::Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
[virtual]

Sends a one-way command. Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3285).

6.504.2.4 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException (transport::Transport * source, const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters:

source The source of the exception
ex The exception.

6.504.2.5 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters:

command The request to send.
timeout The time to wait for the response.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from `activemq::transport::TransportFilter` (p. 3286).

6.504.2.6 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters:

command The request to send.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3287).

6.504.2.7 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::start () throw (decaf::io::IOException) [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands. If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

IOException if an error occurs or if this transport has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3287).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

6.505 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h> Inheritance diagram for activemq::wireformat::openwire::OpenWireResponseBuilder:

Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command > &command**)

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

- virtual void **buildIncomingCommands** (const **Pointer< commands::Command > &command**, **decaf::util::StlQueue< Pointer< commands::Command > > &queue**)

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

6.505.1 Constructor & Destructor Documentation

6.505.1.1 **activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder** () [inline]

6.505.1.2 **virtual** **activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder** () [inline, virtual]

6.505.2 Member Function Documentation

6.505.2.1 **virtual void** **activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands** (const **Pointer< commands::Command > & command**, **decaf::util::StlQueue< Pointer< commands::Command > > & queue**) [virtual]

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters:

command - The Command being sent to the Broker.

queue - Queue of Command sent back from the broker.

Implements **activemq::transport::mock::ResponseBuilder** (p. 2785).

6.505.2.2 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse(const Pointer< commands::Command > & command) [virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters:

command - The command to build a response for

Returns:

A Response object pointer, or NULL if no response.

Implements `activemq::transport::mock::ResponseBuilder` (p. 2786).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h`

6.506 activemq::wireformat::openwire::utils::OpenwireStringSupport Class Reference

```
#include <src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h>
```

Static Public Member Functions

- static std::string **readString** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)

Static method used for reading a string that uses the Openwire format from a DataInputStream, this can throw an IOException for the same reason as a DataInputStream.readUTF might, as well as if the string that is received doesn't conform to the supported character encoding.

- static void **writeString** (decaf::io::DataOutputStream &dataOut, const std::string *str) throw (decaf::io::IOException)

Static method used for writing a string that uses the Openwire format from a DataOutputStream, this can throw an IOException for the same reason as a DataOutputStream.writeUTF might.

Protected Member Functions

- OpenwireStringSupport ()
- virtual ~OpenwireStringSupport ()

6.506.1 Constructor & Destructor Documentation

6.506.1.1 activemq::wireformat::openwire::utils::OpenwireStringSupport::OpenwireStringSupport () [inline, protected]

6.506.1.2 virtual activemq::wireformat::openwire::utils::OpenwireStringSupport::~~OpenwireStringSupport () [inline, protected, virtual]

6.506.2 Member Function Documentation

6.506.2.1 static std::string activemq::wireformat::openwire::utils::OpenwireStringSupport::readString (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException) [static]

Static method used for reading a string that uses the Openwire format from a DataInputStream, this can throw an IOException for the same reason as a DataInputStream.readUTF might, as well as if the string that is received doesn't conform to the supported character encoding.

Parameters:

dataIn - DataInputStream to read from

Returns:

A string that has been read

Exceptions:

IOException on Error.

6.506.2.2 static void activemq::wireformat::openwire::utils::OpenwireStringSupport::writeString (decaf::io::DataOutputStream & *dataOut*, const std::string * *str*) throw (decaf::io::IOException) [static]

Static method used for writing a string that uses the Openwire format from a DataOutputStream, this can throw an IOException for the same reason as a DataOutputStream.writeUTF might.

Parameters:

dataOut - DataOutputStream to write to

str - A pointer to a string that should be written, NULL needs to be an option here as its written as -1.

Exceptions:

IOException on Error.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**OpenwireStringSupport.h**

6.507 decaf::io::OutputStream Class Reference

Base interface for an output stream.

#include <src/main/decaf/io/OutputStream.h> Inheritance diagram for decaf::io::OutputStream:

Public Member Functions

- virtual **~OutputStream** ()
- virtual void **write** (unsigned char c)=0 throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer)=0 throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len)=0 throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** ()=0 throw (IOException)
Flushes any pending writes in this output stream.

6.507.1 Detailed Description

Base interface for an output stream.

6.507.2 Constructor & Destructor Documentation

6.507.2.1 virtual decaf::io::OutputStream::~~OutputStream () [inline, virtual]

6.507.3 Member Function Documentation

6.507.3.1 virtual void decaf::io::OutputStream::flush () throw (IOException)
[pure virtual]

Flushes any pending writes in this output stream.

Exceptions:

IOException (p. 1820)

Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 2995), **decaf::internal::io::StandardOutputStream** (p. 3009), **decaf::io::BufferedOutputStream** (p. 815), **decaf::io::ByteArrayOutputStream** (p. 902), **decaf::io::FilterOutputStream** (p. 1642), and **decaf::net::SocketOutputStream** (p. 2985).

6.507.3.2 virtual void decaf::io::OutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implemented in `activemq::io::LoggingOutputStream` (p. 2040), `decaf::internal::io::StandardErrorOutputStream` (p. 2998), `decaf::internal::io::StandardOutputStream` (p. 3012), `decaf::io::BufferedOutputStream` (p. 815), `decaf::io::ByteArrayOutputStream` (p. 906), `decaf::io::DataOutputStream` (p. 1390), `decaf::io::FilterOutputStream` (p. 1645), and `decaf::net::SocketOutputStream` (p. 2988).

6.507.3.3 virtual void decaf::io::OutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [pure virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implemented in `decaf::internal::io::StandardErrorOutputStream` (p. 2999), `decaf::internal::io::StandardOutputStream` (p. 3012), `decaf::io::BufferedOutputStream` (p. 816), `decaf::io::ByteArrayOutputStream` (p. 906), `decaf::io::DataOutputStream` (p. 1390), `decaf::io::FilterOutputStream` (p. 1646), and `decaf::net::SocketOutputStream` (p. 2989).

6.507.3.4 virtual void decaf::io::OutputStream::write (unsigned char *c*) throw (IOException) [pure virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

Implemented in `activemq::io::LoggingOutputStream` (p. 2041), `decaf::internal::io::StandardErrorOutputStream` (p. 2999), `decaf::internal::io::StandardOutputStream` (p. 3013), `decaf::io::BufferedOutputStream` (p. 816), `decaf::io::ByteArrayOutputStream` (p. 907), `decaf::io::DataOutputStream` (p. 1390), `decaf::io::FilterOutputStream` (p. 1646), and `decaf::net::SocketOutputStream` (p. 2989).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/OutputStream.h`

6.508 activemq::commands::PartialCommand Class Reference

#include <src/main/activemq/commands/PartialCommand.h> Inheritance diagram for activemq::commands::PartialCommand:

Public Member Functions

- **PartialCommand** ()
- virtual **~PartialCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **PartialCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual int **getCommandId** () const
- virtual void **setCommandId** (int commandId)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &data)

Static Public Attributes

- static const unsigned char **ID_PARTIALCOMMAND** = 60

Protected Member Functions

- **PartialCommand** (const **PartialCommand** &)
- **PartialCommand** & **operator=** (const **PartialCommand** &)

Protected Attributes

- int **commandId**
- std::vector< unsigned char > **data**

6.508.1 Constructor & Destructor Documentation

6.508.1.1 `activemq::commands::PartialCommand::PartialCommand (const PartialCommand &) [inline, protected]`

6.508.1.2 `activemq::commands::PartialCommand::PartialCommand ()`

6.508.1.3 `virtual activemq::commands::PartialCommand::~~PartialCommand () [virtual]`

6.508.2 Member Function Documentation

6.508.2.1 `virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 1959).

6.508.2.2 `virtual void activemq::commands::PartialCommand::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 1959).

6.508.2.3 `virtual bool activemq::commands::PartialCommand::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1463).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 1959).

6.508.2.4 `virtual int activemq::commands::PartialCommand::getCommandId () const [virtual]`

6.508.2.5 `virtual std::vector<unsigned char>& activemq::commands::PartialCommand::getData () [virtual]`

6.508.2.6 `virtual const std::vector<unsigned char>& activemq::commands::PartialCommand::getData () const [virtual]`

6.508.2.7 `virtual unsigned char activemq::commands::PartialCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSetructure** (p. 1461) type copy.

Implements **activemq::commands::DataSetructure** (p. 1464).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1960).

6.508.2.8 `PartialCommand& activemq::commands::PartialCommand::operator= (const PartialCommand &) [inline, protected]`

6.508.2.9 `virtual void activemq::commands::PartialCommand::setCommandId (int commandId) [virtual]`

6.508.2.10 `virtual void activemq::commands::PartialCommand::setData (const std::vector< unsigned char > & data) [virtual]`

6.508.2.11 `virtual std::string activemq::commands::PartialCommand::toString () const [virtual]`

Returns a string containing the information for this **DataSetructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSetructure** (p. 718).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1960).

6.508.3 Field Documentation

6.508.3.1 `int activemq::commands::PartialCommand::commandId` [protected]

6.508.3.2 `std::vector<unsigned char> activemq::commands::PartialCommand::data`
[protected]

6.508.3.3 `const unsigned char activemq::commands::PartialCommand::ID_ -`
`PARTIALCOMMAND = 60` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/PartialCommand.h`

6.509 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2477).

#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.509.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2477). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.509.2 Constructor & Destructor Documentation

6.509.2.1 `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

6.509.2.2 `virtual activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

6.509.3 Member Function Documentation

6.509.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 1962).

6.509.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 1962).

6.509.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1962).

6.509.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1963).

6.509.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1963).

6.509.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1448).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 1964).

6.509.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1454).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 1964).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h`

6.510 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2481).

#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.510.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2481). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.510.2 Constructor & Destructor Documentation

6.510.2.1 `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

6.510.2.2 `virtual activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

6.510.3 Member Function Documentation

6.510.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1970).

6.510.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1970).

6.510.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1970).

6.510.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1971).

6.510.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal1
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1971).

6.510.3.6 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1972).

6.510.3.7 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1972).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**PartialCommandMarshaller.h**

6.511 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2485).

#include <src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.511.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2485). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.511.2 Constructor & Destructor Documentation

6.511.2.1 `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

6.511.2.2 `virtual activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

6.511.3 Member Function Documentation

6.511.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1974).

6.511.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1974).

6.511.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1974).

6.511.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1975).

6.511.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal1
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1975).

6.511.3.6 virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1976).

6.511.3.7 virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1976).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**PartialCommandMarshaller.h**

6.512 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2489).

#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.512.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2489). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.512.2 Constructor & Destructor Documentation

6.512.2.1 `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

6.512.2.2 `virtual activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

6.512.3 Member Function Documentation

6.512.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 1966).

6.512.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 1966).

6.512.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1966).

6.512.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1967).

6.512.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal1
(OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, utils::BooleanStream * *bs*) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1967).

6.512.3.6 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1968).

6.512.3.7 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1968).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**PartialCommandMarshaller.h**

6.513 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2493).

#include <src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.513.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2493). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.513.2 Constructor & Destructor Documentation

6.513.2.1 `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::PartialCommandMarshaller()` [inline]

6.513.2.2 `virtual activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::~~PartialCommandMarshaller()` [inline, virtual]

6.513.3 Member Function Documentation

6.513.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1978).

6.513.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1978).

6.513.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1978).

6.513.3.4 virtual void `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1979).

6.513.3.5 virtual int `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1979).

6.513.3.6 virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 1980).

6.513.3.7 virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 1980).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**PartialCommandMarshaller.h**

6.514 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2497) that is a template on a Type and is Thread Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

Public Types

- typedef T * **PointerType**
- typedef T & **ReferenceType**
- typedef REFCOUNTER **CounterType**

Public Member Functions

- **Pointer** ()
Default Constructor.
- **Pointer** (const **PointerType** value)
*Explicit Constructor, creates a **Pointer** (p. 2497) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()
Copy constructor.
- template<typename T1 , typename R1 >
Pointer (const **Pointer**< T1, R1 > &value) throw ()
Copy constructor.
- template<typename T1 , typename R1 >
Pointer (const **Pointer**< T1, R1 > &value, const **STATIC_CAST_TOKEN** &) throw ()
Static Cast constructor.
- template<typename T1 , typename R1 >
Pointer (const **Pointer**< T1, R1 > &value, const **DYNAMIC_CAST_TOKEN** &) throw (decaf::lang::exceptions::ClassCastException)
Dynamic Cast constructor.
- virtual ~**Pointer** () throw ()
- void **reset** (T *value)
*Resets the **Pointer** (p. 2497) to hold the new value.*
- T * **release** ()
*Releases the **Pointer** (p. 2497) held and resets the internal pointer value to Null.*
- **PointerType** **get** () const
*Gets the real pointer that is contained within this **Pointer** (p. 2497).*
- void **swap** (**Pointer** &value) throw ()

Exception (p. 1574) Safe Swap Function.

- **Pointer** & **operator**== (const **Pointer** &right) throw ()
*Assigns the value of right to this **Pointer** (p. 2497) and increments the reference Count.*
- template<typename T1 , typename R1 >
Pointer & **operator**== (const **Pointer**< T1, R1 > &right) throw ()
- **ReferenceType** **operator*** ()
Dereference Operator, returns a reference to the Contained value.
- **ReferenceType** **operator*** () const
- **PointerType** **operator**-> ()
Indirection Operator, returns a pointer to the Contained value.
- **PointerType** **operator**-> () const
- bool **operator**! () const
- template<typename T1 , typename R1 >
 bool **operator**== (const **Pointer**< T1, R1 > &right) const
- template<typename T1 , typename R1 >
 bool **operator**!= (const **Pointer**< T1, R1 > &right) const
- template<typename T1 >
Pointer< T1, **CounterType** > **dynamicCast** () const
- template<typename T1 >
Pointer< T1, **CounterType** > **staticCast** () const

Friends

- bool **operator**== (const **Pointer** &left, const T *right)
- bool **operator**== (const T *left, const **Pointer** &right)
- bool **operator**!= (const **Pointer** &left, const T *right)
- bool **operator**!= (const T *left, const **Pointer** &right)

6.514.1 Detailed Description

```
template<typename T, typename REFCOUNTER = AtomicRefCounter> class
decaf::lang::Pointer< T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2497) that is a template on a Type and is Thread Safe if the default Reference Counter is used. This **Pointer** (p. 2497) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of **ReferenceCounter**.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2497) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2497) in a STL container that requires it, **Pointer** (p. 2497) provides an implementation of std::less.

Since:

1.0

6.514.2 Member Typedef Documentation

- 6.514.2.1** `template<typename T, typename REFCOUNTER = AtomicRefCount> typedef REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER >::CounterType`
- 6.514.2.2** `template<typename T, typename REFCOUNTER = AtomicRefCount> typedef T* decaf::lang::Pointer< T, REFCOUNTER >::PointerType`
- 6.514.2.3** `template<typename T, typename REFCOUNTER = AtomicRefCount> typedef T& decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

6.514.3 Constructor & Destructor Documentation

- 6.514.3.1** `template<typename T, typename REFCOUNTER = AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer () [inline]`

Default Constructor. Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::Pointer< TransactionId >::reset()`.

- 6.514.3.2** `template<typename T, typename REFCOUNTER = AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const PointerType value) [inline, explicit]`

Explicit Constructor, creates a **Pointer** (p. 2497) that contains value with a single reference. This object now has ownership until a call to release.

Parameters:

value - instance of the type we are containing here.

- 6.514.3.3** `template<typename T, typename REFCOUNTER = AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T, REFCOUNTER > & value) throw () [inline]`

Copy constructor. Copies the value contained in the pointer to the new instance and increments the reference counter.

- 6.514.3.4** `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & value) throw () [inline]`

Copy constructor. Copies the value contained in the pointer to the new instance and increments the reference counter.

6.514.3.5 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & value, const STATIC_CAST_TOKEN &) throw () [inline]`

Static Cast constructor. Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p.2497) object.

Parameters:

value - **Pointer** (p.2497) instance to cast to this type.

6.514.3.6 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & value, const DYNAMIC_CAST_TOKEN &) throw (decaf::lang::exceptions::ClassCastException) [inline]`

Dynamic Cast constructor. Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p.2497) object. If the cast fails and return NULL then this method throws a **ClassCastException**.

Parameters:

value - **Pointer** (p.2497) instance to cast to this type.

Exceptions:

ClassCastException if the dynamic cast returns NULL

6.514.3.7 `template<typename T, typename REFCOUNTER = AtomicRefCounter> virtual decaf::lang::Pointer< T, REFCOUNTER >::~Pointer () throw () [inline, virtual]`

6.514.4 Member Function Documentation

6.514.4.1 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::dynamicCast () const [inline]`

6.514.4.2 `template<typename T, typename REFCOUNTER = AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::get () const [inline]`

Gets the real pointer that is contained within this **Pointer** (p.2497). This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p.2497). Use at your own risk.

Returns:

the contained pointer.

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals(), activemq::core::ActiveMQConnectionSupport::getProperties(), activemq::core::ActiveMQConnectionSupport::getTransport(), decaf::lang::operator!=(), decaf::lang::Pointer< TransactionId >::operator!=(), std::less< decaf::lang::Pointer< T > >::operator()(), decaf::lang::operator==(), decaf::lang::Pointer< TransactionId >::operator==(), and activemq::state::ConnectionState::removeTempDestination().

6.514.4.3 `template<typename T, typename REFCOUNTER = AtomicRefCounter> bool decaf::lang::Pointer< T, REFCOUNTER >::operator! () const [inline]`

6.514.4.4 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1, typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator!= (const Pointer< T1, R1 > & right) const [inline]`

6.514.4.5 `template<typename T, typename REFCOUNTER = AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator* () const [inline]`

6.514.4.6 `template<typename T, typename REFCOUNTER = AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator* () [inline]`

Dereference Operator, returns a reference to the Contained value. This method throws an `NullPointerException` if the contained value is `NULL`.

Returns:

reference to the contained pointer.

Exceptions:

NullPointerException if the contained value is `Null`

6.514.4.7 `template<typename T, typename REFCOUNTER = AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> () const [inline]`

6.514.4.8 `template<typename T, typename REFCOUNTER = AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> () [inline]`

Indirection Operator, returns a pointer to the Contained value. This method throws an `NullPointerException` if the contained value is `NULL`.

Returns:

reference to the contained pointer.

Exceptions:

NullPointerException if the contained value is Null

- 6.514.4.9** `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1, typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T1, R1 > & right) throw () [inline]`
- 6.514.4.10** `template<typename T, typename REFCOUNTER = AtomicRefCount> Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T, REFCOUNTER > & right) throw () [inline]`

Assigns the value of *right* to this **Pointer** (p. 2497) and increments the reference Count.

Parameters:

right - **Pointer** (p. 2497) on the right hand side of an operator= call to this.

- 6.514.4.11** `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1, typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (const Pointer< T1, R1 > & right) const [inline]`
- 6.514.4.12** `template<typename T, typename REFCOUNTER = AtomicRefCount> T* decaf::lang::Pointer< T, REFCOUNTER >::release () [inline]`

Releases the **Pointer** (p. 2497) held and resets the internal pointer value to Null. This method is not guaranteed to be safe if the **Pointer** (p. 2497) is held by more than one object or this method is called from more than one thread.

Parameters:

value - The new value to contain.

Returns:

The pointer instance that was held by this **Pointer** (p. 2497) object, the pointer is no longer owned by this **Pointer** (p. 2497) and won't be freed when this **Pointer** (p. 2497) goes out of scope.

Referenced by `decaf::lang::Pointer< TransactionId >::Pointer()`.

- 6.514.4.13** `template<typename T, typename REFCOUNTER = AtomicRefCount> void decaf::lang::Pointer< T, REFCOUNTER >::reset (T * value) [inline]`

Resets the **Pointer** (p. 2497) to hold the new value. Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2497) to a NULL pointer.

Parameters:

value - The new value to contain.

6.514.4.14 `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::staticCast () const [inline]`

6.514.4.15 `template<typename T, typename REFCOUNTER = AtomicRefCount> void decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T, REFCOUNTER > & value) throw () [inline]`

Exception (p.1574) Safe Swap Function.

Parameters:

value - the value to swap with this.

Referenced by `decaf::lang::Pointer< TransactionId >::operator=()`, and `decaf::lang::Pointer< TransactionId >::swap()`.

6.514.5 Friends And Related Function Documentation

6.514.5.1 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator!= (const T * left, const Pointer< T, REFCOUNTER > & right) [friend]`

6.514.5.2 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator!= (const Pointer< T, REFCOUNTER > & left, const T * right) [friend]`

6.514.5.3 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator== (const T * left, const Pointer< T, REFCOUNTER > & right) [friend]`

6.514.5.4 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator== (const Pointer< T, REFCOUNTER > & left, const T * right) [friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.515 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2497) instance.

#include <src/main/decaf/lang/Pointer.h> Inheritance diagram for decaf::lang::PointerComparator< T, R >:

Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

6.515.1 Detailed Description

template<typename T, typename R = AtomicRefCounter> class decaf::lang::PointerComparator< T, R >

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2497) instance. This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2497) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

6.515.2 Member Function Documentation

6.515.2.1 template<typename T, typename R = AtomicRefCounter> virtual int decaf::lang::PointerComparator< T, R >::compare (const **Pointer**< T, R > & *left*, const **Pointer**< T, R > & *right*) const [inline, virtual]

6.515.2.2 template<typename T, typename R = AtomicRefCounter> virtual bool decaf::lang::PointerComparator< T, R >::operator() (const **Pointer**< T, R > & *left*, const **Pointer**< T, R > & *right*) const [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.516 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

#include <src/main/activemq/cmsutil/PooledSession.h> Inheritance diagram for activemq::cmsutil::PooledSession:

Public Member Functions

- **PooledSession** (**SessionPool** *pool, **cms::Session** *session)
- virtual **~PooledSession** ()
Does nothing.
- virtual **cms::Session** * **getSession** ()
Returns a non-constant reference to the internal session object.
- virtual const **cms::Session** * **getSession** () const
Returns a constant reference to the internal session object.
- virtual void **close** () throw (cms::CMSEException)
Returns this session back to the pool, but does not close or destroy the internal session object.
- virtual void **commit** () throw (cms::CMSEException)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** () throw (cms::CMSEException)
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** () throw (cms::CMSEException)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSEException)
Creates a durable subscriber to the specified topic, using a Message selector.

- virtual **cms::MessageConsumer** * **createCachedConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.
- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination) throw (cms::CMSEException)
Creates a MessageProducer to send messages to the specified destination.
- virtual **cms::MessageProducer** * **createCachedProducer** (const **cms::Destination** *destination) throw (cms::CMSEException)
First checks the internal producer cache and creates one if none exist for the given destination.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSEException)
Creates a queue identity given a Queue name.
- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSEException)
Creates a topic identity given a Queue name.
- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSEException)
Creates a TemporaryQueue object.
- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSEException)
Creates a TemporaryTopic object.
- virtual **cms::Message** * **createMessage** () throw (cms::CMSEException)
Creates a new Message.
- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSEException)
Creates a BytesMessage.
- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize) throw (cms::CMSEException)
Creates a BytesMessage and sets the payload to the passed value.
- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSEException)
Creates a new StreamMessage.

- virtual **cms::TextMessage** * **createTextMessage** () throw (cms::CMSEException)
Creates a new TextMessage.
- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text) throw (cms::CMSEException)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage** * **createMapMessage** () throw (cms::CMSEException)
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw (cms::CMSEException)
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const throw (cms::CMSEException)
Gets if the Sessions is a Transacted Session.
- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSEException)
Unsubscribes a durable subscription that has been created by a client.

6.516.1 Detailed Description

A pooled session object that wraps around a delegate session. Calls to close this session only result in giving the session back to the pool.

6.516.2 Constructor & Destructor Documentation

6.516.2.1 **activemq::cmsutil::PooledSession::PooledSession** (SessionPool * *pool*, cms::Session * *session*)

6.516.2.2 **virtual activemq::cmsutil::PooledSession::~~PooledSession** () [virtual]

Does nothing.

6.516.3 Member Function Documentation

6.516.3.1 **virtual void activemq::cmsutil::PooledSession::close** () throw (cms::CMSEException) [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object. Implements **cms::Session** (p. 2842).

6.516.3.2 **virtual void activemq::cmsutil::PooledSession::commit** () throw (cms::CMSEException) [inline, virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSEException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements **cms::Session** (p. 2843).

References cms::Session::commit().

6.516.3.3 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw (cms::CMSEException)` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2843).

6.516.3.4 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue) throw (cms::CMSEException)` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2843).

6.516.3.5 `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * bytes, std::size_t bytesSize) throw (cms::CMSException) [inline, virtual]`

Creates a BytesMessage and sets the payload to the passed value.

Parameters:

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2844).

References cms::Session::createBytesMessage().

6.516.3.6 `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage () throw (cms::CMSException) [inline, virtual]`

Creates a BytesMessage.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2844).

References cms::Session::createBytesMessage().

6.516.3.7 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSException) [virtual]`

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal. If created, the consumer is added to the pool's lifecycle manager.

Parameters:

destination the destination to receive on

selector the selector to use

noLocal whether or not to receive messages from the same connection

Returns:

the consumer resource

Exceptions:

cms::CMSException (p. 1031) if something goes wrong.

6.516.3.8 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer (const cms::Destination * destination) throw (cms::CMSEException)` [virtual]

First checks the internal producer cache and creates one if none exist for the given destination. If created, the producer is added to the pool's lifecycle manager.

Parameters:

destination the destination to send on

Returns:

the producer resource

Exceptions:

cms::CMSEException (p. 1031) if something goes wrong.

6.516.3.9 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSEException)` [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2844).

References `cms::Session::createConsumer()`.

6.516.3.10 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw (cms::CMSEException)` [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

Returns:

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2845).

References cms::Session::createConsumer().

```
6.516.3.11 virtual cms::MessageConsumer* ac-
tivemq::cmsutil::PooledSession::createConsumer (const
cms::Destination * destination) throw ( cms::CMSEException ) [inline,
virtual]
```

Creates a MessageConsumer for the specified destination.

Parameters:

destination the Destination that this consumer receiving messages for.

Returns:

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 2845).

References cms::Session::createConsumer().

```
6.516.3.12 virtual cms::MessageConsumer* ac-
tivemq::cmsutil::PooledSession::createDurableConsumer
(const cms::Topic * destination, const std::string & name,
const std::string & selector, bool noLocal = false) throw (
cms::CMSEException ) [inline, virtual]
```

Creates a durable subscriber to the specified topic, using a Message selector. Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters:

destination the topic to subscribe to

name The name used to identify the subscription

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new durable MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2846).

References cms::Session::createDurableConsumer().

6.516.3.13 `virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage ()
throw (cms::CMSEException) [inline, virtual]`

Creates a new MapMessage.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2846).

References cms::Session::createMapMessage().

6.516.3.14 `virtual cms::Message* activemq::cmsutil::PooledSession::createMessage
() throw (cms::CMSEException) [inline, virtual]`

Creates a new Message.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2847).

References cms::Session::createMessage().

6.516.3.15 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer (const
cms::Destination * destination) throw (cms::CMSEException) [inline,
virtual]`

Creates a MessageProducer to send messages to the specified destination.

Parameters:

destination the Destination to send on

Returns:

New MessageProducer that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 2847).

References cms::Session::createProducer().

6.516.3.16 `virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue
(const std::string & queueName) throw (cms::CMSEException)
[inline, virtual]`

Creates a queue identity given a Queue name.

Parameters:

queueName the name of the new Queue

Returns:

new Queue pointer that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2847).

References cms::Session::createQueue().

6.516.3.17 `virtual cms::StreamMessage* ac-
tivemq::cmsutil::PooledSession::createStreamMessage ()
throw (cms::CMSEException) [inline, virtual]`

Creates a new StreamMessage.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2848).

References cms::Session::createStreamMessage().

6.516.3.18 `virtual cms::TemporaryQueue* activemq::cmsutil::PooledSession::createTemporaryQueue ()
throw (cms::CMSEException) [inline, virtual]`

Creates a TemporaryQueue object.

Returns:

new TemporaryQueue pointer that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2848).

References cms::Session::createTemporaryQueue().

6.516.3.19 `virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic ()
throw (cms::CMSEException) [inline, virtual]`

Creates a TemporaryTopic object.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2848).

References cms::Session::createTemporaryTopic().

6.516.3.20 `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage (const
std::string & text) throw (cms::CMSEException) [inline, virtual]`

Creates a new TextMessage and set the text to the value given.

Parameters:

text the initial text for the message

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2848).

References cms::Session::createTextMessage().

6.516.3.21 `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage ()
throw (cms::CMSEException) [inline, virtual]`

Creates a new TextMessage.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2849).

References cms::Session::createTextMessage().

6.516.3.22 `virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic (const std::string & topicName) throw (cms::CMSEException) [inline, virtual]`

Creates a topic identity given a Queue name.

Parameters:

topicName the name of the new Topic

Returns:

new Topic pointer that is owned by the caller.

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2849).

References cms::Session::createTopic().

6.516.3.23 `virtual cms::Session::AcknowledgeMode activemq::cmsutil::PooledSession::getAcknowledgeMode () const throw (cms::CMSEException) [inline, virtual]`

Returns the acknowledgment mode of the session.

Returns:

the Sessions Acknowledge Mode

Exceptions:

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2849).

References cms::Session::getAcknowledgeMode().

6.516.3.24 `virtual const cms::Session* activemq::cmsutil::PooledSession::getSession () const [inline, virtual]`

Returns a constant reference to the internal session object.

Returns:

the session object.

6.516.3.25 `virtual cms::Session* activemq::cmsutil::PooledSession::getSession ()`
[inline, virtual]

Returns a non-constant reference to the internal session object.

Returns:

the session object.

6.516.3.26 `virtual bool activemq::cmsutil::PooledSession::isTransacted () const`
`throw (cms::CMSException)` [inline, virtual]

Gets if the Sessions is a Transacted Session.

Returns:

transacted true - false.

Exceptions:

CMSException - If an internal error occurs.

Implements `cms::Session` (p. 2850).

References `cms::Session::isTransacted()`.

6.516.3.27 `virtual void activemq::cmsutil::PooledSession::recover () throw (`
`cms::CMSException)` [inline, virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions:

CMSException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

Implements `cms::Session` (p. 2850).

References `cms::Session::recover()`.

6.516.3.28 `virtual void activemq::cmsutil::PooledSession::rollback () throw (cms::CMSException) [inline, virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements **cms::Session** (p. 2850).

References `cms::Session::rollback()`.

6.516.3.29 `virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & name) throw (cms::CMSException) [inline, virtual]`

Unsubscribes a durable subscription that has been created by a client. This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters:

name The name used to identify this subscription

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2851).

References `cms::Session::unsubscribe()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/PooledSession.h`

6.517 decaf::util::concurrent::PooledThread Class Reference

```
#include <src/main/decaf/util/concurrent/PooledThread.h>
```

Public Member Functions

- **PooledThread** (**ThreadPool** *pool)
Constructor.
- virtual **~PooledThread** ()
- virtual void **run** ()
*Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3175) and then grabs it and calls its run method.*
- virtual void **stop** () throw (lang::Exception)
Stops the Thread, thread will complete its task if currently running one, and then die.
- virtual bool **isBusy** ()
Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.
- virtual void **setPooledThreadListener** (**PooledThreadListener** *listener)
*Adds a listener to this **PooledThread** (p. 2518) to be notified when this thread starts and completes a task.*
- virtual **PooledThreadListener** * **getPooledThreadListener** ()
*Removes a listener for this **PooledThread** (p. 2518) to be notified when this thread starts and completes a task.*

6.517.1 Constructor & Destructor Documentation

6.517.1.1 decaf::util::concurrent::PooledThread::PooledThread (**ThreadPool** * pool)

Constructor.

Parameters:

pool the parant **ThreadPool** (p. 3175) object

6.517.1.2 virtual decaf::util::concurrent::PooledThread::~~PooledThread () [virtual]

6.517.2 Member Function Documentation

6.517.2.1 virtual **PooledThreadListener*** decaf::util::concurrent::PooledThread::getPooledThreadListener () [inline, virtual]

Removes a listener for this **PooledThread** (p. 2518) to be notified when this thread starts and completes a task.

Returns:

a pointer to this thread's listener or NULL

6.517.2.2 virtual bool decaf::util::concurrent::PooledThread::isBusy () [inline, virtual]

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.

Returns:

true if the Thread is busy

6.517.2.3 virtual void decaf::util::concurrent::PooledThread::run () [virtual]

Run Method for this object waits for something to be enqueued on the **ThreadPool** (p.3175) and then grabs it and calls its run method.

6.517.2.4 virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener (PooledThreadListener * *listener*) [inline, virtual]

Adds a listener to this PooledThread (p.2518) to be notified when this thread starts and completes a task.

Parameters:

listener the listener to send notifications to.

6.517.2.5 virtual void decaf::util::concurrent::PooledThread::stop () throw (lang::Exception) [virtual]

Stops the Thread, thread will complete its task if currently running one, and then die. Does not block.

Exceptions:

Exception

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThread.h**

6.518 decaf::util::concurrent::PooledThreadListener Class Reference

Abstract Listener Interface for users of `ThreadPool` (p. 3175).

#include <src/main/decaf/util/concurrent/PooledThreadListener.h> Inheritance diagram for decaf::util::concurrent::PooledThreadListener:

Public Member Functions

- virtual `~PooledThreadListener ()`
- virtual void `onTaskStarted (PooledThread *thread)=0`
Called by a pooled thread when it is about to begin executing a new task.
- virtual void `onTaskCompleted (PooledThread *thread)=0`
Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.
- virtual void `onTaskException (PooledThread *thread, lang::Exception &ex)=0`
*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2518) is now no longer running.*

6.518.1 Detailed Description

Abstract Listener Interface for users of `ThreadPool` (p. 3175). The implementor of this class receives events related to the execution and termination of threads running in the **ThreadPool** (p. 3175).

Since:

1.0

6.518.2 Constructor & Destructor Documentation

- 6.518.2.1** virtual
`decaf::util::concurrent::PooledThreadListener::~~PooledThreadListener ()`
 [inline, virtual]

6.518.3 Member Function Documentation

- 6.518.3.1** virtual void de-
`caf::util::concurrent::PooledThreadListener::onTaskCompleted`
`(PooledThread * thread)` [pure virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

Parameters:

thread - Pointer the the Pooled Thread that is making this call.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3178).

6.518.3.2 virtual void decaf::util::concurrent::PooledThreadListener::onTaskException
(PooledThread * *thread*, lang::Exception & *ex*) [pure virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p.2518) is now no longer running.

Parameters:

thread - Pointer to the Pooled Thread that is making this call

ex - The Exception that occurred.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3178).

6.518.3.3 virtual void decaf::util::concurrent::PooledThreadListener::onTaskStarted
(PooledThread * *thread*) [pure virtual]

Called by a pooled thread when it is about to begin executing a new task.

Parameters:

thread - Pointer to the Pooled Thread that is making this call

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3179).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThreadListener.h**

6.519 decaf::net::PortUnreachableException Class Reference

#include <src/main/decaf/net/PortUnreachableException.h> Inheritance diagram for decaf::net::PortUnreachableException:

Public Member Functions

- **PortUnreachableException** () throw ()
Default Constructor.
- **PortUnreachableException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()
Copy Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **PortUnreachableException** (const std::exception *cause) throw ()
Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **PortUnreachableException** * clone () const
Clones this exception.
- virtual ~**PortUnreachableException** () throw ()

6.519.1 Constructor & Destructor Documentation

6.519.1.1 decaf::net::PortUnreachableException::PortUnreachableException () throw () [inline]

Default Constructor.

6.519.1.2 decaf::net::PortUnreachableException::PortUnreachableException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.519.1.3 decaf::net::PortUnreachableException::PortUnreachableException (const PortUnreachableException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.519.1.4 decaf::net::PortUnreachableException::PortUnreachableException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.519.1.5 decaf::net::PortUnreachableException::PortUnreachableException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.519.1.6 decaf::net::PortUnreachableException::PortUnreachableException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.519.1.7 **virtual**
decaf::net::PortUnreachableException::~~PortUnreachableException ()
throw () [inline, virtual]

6.519.2 Member Function Documentation

6.519.2.1 **virtual PortUnreachableException* de-**
caf::net::PortUnreachableException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2973).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**PortUnreachableException.h**

6.520 activemq::util::PrimitiveList Class Reference

List of primitives.

#include <src/main/activemq/util/PrimitiveList.h> Inheritance diagram for activemq::util::PrimitiveList:

Public Member Functions

- **PrimitiveList** ()
Default Constructor, creates an Empty list.
- virtual **~PrimitiveList** ()
- **PrimitiveList** (const **decaf::util::List**< **PrimitiveValueNode** > &src)
Copy Constructor.
- **PrimitiveList** (const **PrimitiveList** &src)
Copy Constructor.
- std::string **toString** () const
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual bool **getBool** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Boolean value at the specified index.
- virtual void **setBool** (std::size_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual unsigned char **getByte** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Byte value at the specified index.
- virtual void **setByte** (std::size_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual char **getChar** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Character value at the specified index.

- virtual void **setChar** (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual short **getShort** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Short value at the specified index.
- virtual void **setShort** (std::size_t index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual int **getInt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Integer value at the specified index.
- virtual void **setInt** (std::size_t index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual long long **getLong** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Long value at the specified index.
- virtual void **setLong** (std::size_t index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual float **getFloat** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Float value at the specified index.
- virtual void **setFloat** (std::size_t index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.
- virtual double **getDouble** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the specified index.

- virtual void **setDouble** (std::size_t index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::string **getString** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the String value at the specified index.

- virtual void **setString** (std::size_t index, const std::string &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::vector< unsigned char > **getByteArray** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Byte Array value at the specified index.

- virtual void **setByteArray** (std::size_t index, const std::vector< unsigned char > &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

6.520.1 Detailed Description

List of primitives.

6.520.2 Constructor & Destructor Documentation

6.520.2.1 activemq::util::PrimitiveList::PrimitiveList ()

Default Constructor, creates an Empty list.

6.520.2.2 virtual activemq::util::PrimitiveList::~~PrimitiveList () [virtual]

6.520.2.3 activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & src)

Copy Constructor.

Parameters:

src - the Decaf List of PrimitiveNodeValues to copy

6.520.2.4 `activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & src)`

Copy Constructor.

Parameters:

src - the `PrimitiveList` (p. 2525) to copy

6.520.3 Member Function Documentation

6.520.3.1 `virtual bool activemq::util::PrimitiveList::getBool (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Boolean value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.2 `virtual unsigned char activemq::util::PrimitiveList::getBytes (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.3 `virtual std::vector<unsigned char> activemq::util::PrimitiveList::getByteArray (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte Array value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.4 `virtual char activemq::util::PrimitiveList::getChar (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Character value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.5 `virtual double activemq::util::PrimitiveList::getDouble (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Double value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.6 **virtual float activemq::util::PrimitiveList::getFloat (std::size_t *index*)**
 const throw (decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Float value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.7 **virtual int activemq::util::PrimitiveList::getInt (std::size_t *index*)**
 const throw (decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Integer value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.8 **virtual long long activemq::util::PrimitiveList::getLong (std::size_t *index*)**
 const throw (decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Long value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.9 virtual short activemq::util::PrimitiveList::getShort (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Short value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.10 virtual std::string activemq::util::PrimitiveList::getString (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the String value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > `size()` (p. 3028)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.11 `virtual void activemq::util::PrimitiveList::setBool (std::size_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if `index > size()` (p. 3028).

6.520.3.12 `virtual void activemq::util::PrimitiveList::setByte (std::size_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if `index > size()` (p. 3028).

6.520.3.13 `virtual void activemq::util::PrimitiveList::setByteArray (std::size_t index, const std::vector< unsigned char > & value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if `index > size()` (p. 3028).

6.520.3.14 **virtual void activemq::util::PrimitiveList::setChar (std::size_t *index*, char *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException)** [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > size() (p. 3028).

6.520.3.15 **virtual void activemq::util::PrimitiveList::setDouble (std::size_t *index*, double *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException)** [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > size() (p. 3028).

6.520.3.16 **virtual void activemq::util::PrimitiveList::setFloat (std::size_t *index*, float *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException)** [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > size() (p. 3028).

6.520.3.17 `virtual void activemq::util::PrimitiveList::setInt (std::size_t index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > size() (p. 3028).

6.520.3.18 `virtual void activemq::util::PrimitiveList::setLong (std::size_t index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > size() (p. 3028).

6.520.3.19 `virtual void activemq::util::PrimitiveList::setShort (std::size_t index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > size() (p. 3028).

6.520.3.20 `virtual void activemq::util::PrimitiveList::setString
 (std::size_t index, const std::string & value) throw (
 decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if `index > size()` (p. 3028).

6.520.3.21 `std::string activemq::util::PrimitiveList::toString () const`

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns:

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveList.h`

6.521 activemq::util::PrimitiveMap Class Reference

Map of named primitives.

#include <src/main/activemq/util/PrimitiveMap.h> Inheritance diagram for activemq::util::PrimitiveMap:

Public Member Functions

- **PrimitiveMap** ()
Default Constructor, creates an empty map.
- virtual **~PrimitiveMap** ()
- **PrimitiveMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &source)
Copy Constructor.
- **PrimitiveMap** (const **PrimitiveMap** &source)
Copy Constructor.
- std::string **toString** () const
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual bool **getBool** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setBool** (const std::string &key, bool value)
Sets the value at key to the specified type.
- virtual unsigned char **getByte** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setByte** (const std::string &key, unsigned char value)
Sets the value at key to the specified type.
- virtual char **getChar** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)
Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.
- virtual void **setChar** (const std::string &key, char value)

Sets the value at key to the specified type.

- virtual short **getShort** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setShort** (const std::string &key, short value)

Sets the value at key to the specified type.

- virtual int **getInt** (const std::string &key) const throw (de-
cafe::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setInt** (const std::string &key, int value)

Sets the value at key to the specified type.

- virtual long long **getLong** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setLong** (const std::string &key, long long value)

Sets the value at key to the specified type.

- virtual float **getFloat** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setFloat** (const std::string &key, float value)

Sets the value at key to the specified type.

- virtual double **getDouble** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setDouble** (const std::string &key, double value)

Sets the value at key to the specified type.

- virtual std::string **getString** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
cafe::lang::exceptions::UnsupportedOperationException)

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setString** (const std::string &key, const std::string &value)

Sets the value at key to the specified type.

- virtual std::vector< unsigned char > **getByteArray** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char > &value)

Sets the value at key to the specified type.

6.521.1 Detailed Description

Map of named primitives.

6.521.2 Constructor & Destructor Documentation

6.521.2.1 activemq::util::PrimitiveMap::PrimitiveMap ()

Default Constructor, creates an empty map.

6.521.2.2 virtual activemq::util::PrimitiveMap::~~PrimitiveMap () [virtual]

6.521.2.3 activemq::util::PrimitiveMap::PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > & source)

Copy Constructor.

Parameters:

source The Decaf Library Map instance whose elements will be copied into this Map.

6.521.2.4 activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & source)

Copy Constructor.

Parameters:

source The **PrimitiveMap** (p. 2536) whose elements will be copied into this Map.

6.521.3 Member Function Documentation

6.521.3.1 `virtual bool activemq::util::PrimitiveMap::getBool (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.2 `virtual unsigned char activemq::util::PrimitiveMap::getBytes (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.3 `virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getBytesArray (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at *key* in the type requested.

Exceptions:

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.4 `virtual char activemq::util::PrimitiveMap::getChar (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Character value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at *key* in the type requested.

Exceptions:

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.5 `virtual double activemq::util::PrimitiveMap::getDouble (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Double value at the given *key*, if the *key* is not in the map or cannot be returned as the requested value then an exception of type *NoSuchElementException* is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at *key* in the type requested.

Exceptions:

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.6 **virtual float activemq::util::PrimitiveMap::getFloat (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)** [virtual]

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.7 **virtual int activemq::util::PrimitiveMap::getInt (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)** [virtual]

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.8 **virtual long long activemq::util::PrimitiveMap::getLong (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)** [virtual]

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at *key* in the type requested.

Exceptions:

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.9 virtual short activemq::util::PrimitiveMap::getShort (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at *key* in the type requested.

Exceptions:

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.10 virtual std::string activemq::util::PrimitiveMap::getString (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at *key* in the type requested.

Exceptions:

NoSuchElementException if *key* is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.11 virtual void activemq::util::PrimitiveMap::setBool (const std::string & *key*, bool *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.12 virtual void activemq::util::PrimitiveMap::setByte (const std::string & *key*, unsigned char *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.13 virtual void activemq::util::PrimitiveMap::setByteArray (const std::string & *key*, const std::vector< unsigned char > & *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.14 virtual void activemq::util::PrimitiveMap::setChar (const std::string & *key*, char *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.15 virtual void activemq::util::PrimitiveMap::setDouble (const std::string & *key*, double *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.16 virtual void activemq::util::PrimitiveMap::setFloat (const std::string & *key*, float *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.17 virtual void activemq::util::PrimitiveMap::setInt (const std::string & *key*, int *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.18 virtual void activemq::util::PrimitiveMap::setLong (const std::string & *key*, long long *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.19 virtual void activemq::util::PrimitiveMap::setShort (const std::string & *key*, short *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.20 virtual void activemq::util::PrimitiveMap::setString (const std::string & *key*, const std::string & *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

- key* - the map key to set or insert.
- value* - the new value to set at the key location.

6.521.3.21 std::string activemq::util::PrimitiveMap::toString () const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns:

- formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveMap.h**

6.522 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>
```

Public Member Functions

- **PrimitiveTypesMarshaller** ()
- virtual **~PrimitiveTypesMarshaller** ()

Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** *map, std::vector< unsigned char > &dest) throw (decaf::lang::Exception)
Static Marshal of a primitive map object.
- static void **unmarshal** (**util::PrimitiveMap** *map, const std::vector< unsigned char > &src) throw (decaf::lang::Exception)
Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.
- static void **marshal** (const **util::PrimitiveList** *list, std::vector< unsigned char > &dest) throw (decaf::lang::Exception)
Static Marshal of a primitive map object.
- static void **unmarshal** (**util::PrimitiveList** *list, const std::vector< unsigned char > &src) throw (decaf::lang::Exception)
Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.

Static Protected Member Functions

- static void **marshalPrimitiveMap** (decaf::io::DataOutputStream &dataOut, const decaf::util::Map< std::string, util::PrimitiveValueNode > &map) throw (decaf::io::IOException)
Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.
- static void **marshalPrimitiveList** (decaf::io::DataOutputStream &dataOut, const decaf::util::List< util::PrimitiveValueNode > &list) throw (decaf::io::IOException)
Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.
- static void **marshalPrimitive** (decaf::io::DataOutputStream &dataOut, const util::PrimitiveValueNode &value) throw (decaf::io::IOException)
Used to Marshal the Primitive types out on the Wire.

- static void **unmarshalPrimitiveMap** (decaf::io::DataInputStream &dataIn, util::PrimitiveMap &map) throw (decaf::io::IOException)
Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.
- static void **unmarshalPrimitiveList** (decaf::io::DataInputStream &dataIn, decaf::util::StIList< util::PrimitiveValueNode > &list) throw (decaf::io::IOException)
Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.
- static util::PrimitiveValueNode **unmarshalPrimitive** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)
Unmarshals a Primitive Type from the stream, and returns it as a value Node.

6.522.1 Detailed Description

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

6.522.2 Constructor & Destructor Documentation

- 6.522.2.1** **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller**
() [inline]
- 6.522.2.2** **virtual**
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller
() [inline, virtual]

6.522.3 Member Function Documentation

- 6.522.3.1** static void **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal**
(const util::PrimitiveList * *list*, std::vector< unsigned char > & *dest*)
throw (decaf::lang::Exception) [static]

Static Marshal of a primitive map object.

Parameters:

- list* The list object to Marshal
- dest* Reference to a byte array to house the data

Exceptions:

Exception

```
6.522.3.2 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal
(const util::PrimitiveMap * map, std::vector< unsigned char > & dest)
throw ( decaf::lang::Exception ) [static]
```

Static Marshal of a primitive map object.

Parameters:

map Map to Marshal.

dest Reference to a byte array to house the data.

Exceptions:

Exception

```
6.522.3.3 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive
(decaf::io::DataOutputStream & dataOut, const util::PrimitiveValueNode
& value) throw ( decaf::io::IOException ) [static, protected]
```

Used to Marshal the Primitive types out on the Wire.

Parameters:

dataOut - the DataOutputStream to write to

value - the ValueNode to write.

Exceptions:

IOException

```
6.522.3.4 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList
(decaf::io::DataOutputStream & dataOut, const decaf::util::List<
util::PrimitiveValueNode > & list) throw ( decaf::io::IOException )
[static, protected]
```

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters:

dataOut - the DataOutputStream to write to

list - the ValueNode to write.

Exceptions:

IOException

6.522.3.5 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap (decaf::io::DataOutputStream & *dataOut*, const decaf::util::Map< std::string, util::PrimitiveValueNode > & *map*) throw (decaf::io::IOException) [static, protected]

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters:

dataOut - the DataOutputStream to write to
map - the ValueNode to write.

Exceptions:

IOException

6.522.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveList * *list*, const std::vector< unsigned char > & *src*) throw (decaf::lang::Exception) [static]

Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object. Caller owns the pointer.

Parameters:

list The list object to Un-marshal
src Reference to a byte array to read data from.

Exceptions:

Exception

6.522.3.7 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveMap * *map*, const std::vector< unsigned char > & *src*) throw (decaf::lang::Exception) [static]

Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object. Caller owns the pointer.

Parameters:

map Map to Unmarshal into
src Reference to a byte array to read data from.

Exceptions:

Exception

6.522.3.8 static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException)
[static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

Parameters:

dataIn - DataInputStream to read from.

Returns:

a PrimitiveValueNode containing the data.

Exceptions:

IOException

6.522.3.9 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList (decaf::io::DataInputStream & *dataIn*, decaf::util::StlList< util::PrimitiveValueNode > & *list*) throw (decaf::io::IOException)
[static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters:

dataIn - DataInputStream to read from.

list - the ValueNode to write.

Exceptions:

IOException

6.522.3.10 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap (decaf::io::DataInputStream & *dataIn*, util::PrimitiveMap & *map*)
throw (decaf::io::IOException) [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters:

dataIn - DataInputStream to read from.

map - the map to fill with data.

Exceptions:

IOException

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

6.523 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string * **stringValue**
- std::vector< unsigned char > * **byteArrayValue**
- decaf::util::List< PrimitiveValueNode > * **listValue**
- decaf::util::Map< std::string, PrimitiveValueNode > * **mapValue**

6.523.1 Detailed Description

Define a union type comprised of the various types.

6.523.2 Field Documentation

- 6.523.2.1 `bool activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue`
- 6.523.2.2 `std::vector<unsigned char>* activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue`
- 6.523.2.3 `unsigned char activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue`
- 6.523.2.4 `char activemq::util::PrimitiveValueNode::PrimitiveValue::charValue`
- 6.523.2.5 `double activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue`
- 6.523.2.6 `float activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue`
- 6.523.2.7 `int activemq::util::PrimitiveValueNode::PrimitiveValue::intValue`
- 6.523.2.8 `decaf::util::List<PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::listValue`
- 6.523.2.9 `long long activemq::util::PrimitiveValueNode::PrimitiveValue::longValue`
- 6.523.2.10 `decaf::util::Map<std::string, PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue`
- 6.523.2.11 `short activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue`
- 6.523.2.12 `std::string* activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue`

The documentation for this union was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

6.524 activemq::util::PrimitiveValueConverter Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p.2555) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- template<typename TO >
TO **convert** (const **PrimitiveValueNode** &value) const throw (decaf::lang::exceptions::UnsupportedOperationException)

6.524.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p.2555) from one type to another. If the conversion is supported then calling the convert method will throw an UnsupportedOperationException to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X X							
byte		X X X X X						
short			X X X X					
int				X X X X				
long					X X			
float						X X X		
double							X X X	
String								X X X X X X X X

Since:

3.0

6.524.2 Constructor & Destructor Documentation

6.524.2.1 **activemq::util::PrimitiveValueConverter::PrimitiveValueConverter** ()
[inline]

6.524.2.2 **virtual**
activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter ()
[inline, virtual]

6.524.3 Member Function Documentation

6.524.3.1 **std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert** (const **PrimitiveValueNode** & *value*) const throw (decaf::lang::exceptions::UnsupportedOperationException)
[inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueConverter.h`

6.525 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Structures

- union **PrimitiveValue**

Define a union type comprised of the various types.

Public Types

- enum **PrimitiveType** {
 NULL_TYPE = 0, **BOOLEAN_TYPE** = 1, **BYTE_TYPE** = 2, **CHAR_TYPE** = 3,
 SHORT_TYPE = 4, **INTEGER_TYPE** = 5, **LONG_TYPE** = 6, **DOUBLE_TYPE** = 7,
 FLOAT_TYPE = 8, **STRING_TYPE** = 9, **BYTE_ARRAY_TYPE** = 10, **MAP_TYPE** = 11,
 LIST_TYPE = 12, **BIG_STRING_TYPE** = 13 }

Enumeration for the various primitive types.

Public Member Functions

- **PrimitiveValueNode** ()
Default Constructor, creates a value of the NULL_TYPE.
- **PrimitiveValueNode** (bool value)
Boolean Value Constructor.
- **PrimitiveValueNode** (unsigned char value)
Byte Value Constructor.
- **PrimitiveValueNode** (char value)
Char Value Constructor.
- **PrimitiveValueNode** (short value)
Short Value Constructor.
- **PrimitiveValueNode** (int value)
Int Value Constructor.
- **PrimitiveValueNode** (long long value)
Long Value Constructor.
- **PrimitiveValueNode** (float value)

Float Value Constructor.

- **PrimitiveValueNode** (double value)
Double Value Constructor.
- **PrimitiveValueNode** (const char *value)
String Value Constructor.
- **PrimitiveValueNode** (const std::string &value)
String Value Constructor.
- **PrimitiveValueNode** (const std::vector< unsigned char > &value)
Byte Array Value Constructor.
- **PrimitiveValueNode** (const decaf::util::List< PrimitiveValueNode > &value)
Primitive List Constructor.
- **PrimitiveValueNode** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)
Primitive Map Value Constructor.
- **PrimitiveValueNode** (const PrimitiveValueNode &node)
Copy constructor.
- **~PrimitiveValueNode** ()
- **PrimitiveValueNode & operator=** (const PrimitiveValueNode &node)
Assignment operator, copies the data from the other node.
- **bool operator==** (const PrimitiveValueNode &node) const
Comparison Operator, compares this node to the other node.
- **PrimitiveType getType** () const
Gets the Value Type of this type wrapper.
- **PrimitiveValue getValue** () const
Gets the internal Primitive Value object from this wrapper.
- **void setValue** (const PrimitiveValue &value, PrimitiveType valueType)
Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.
- **void clear** ()
Clears the value from this wrapper converting it back to a blank NULL_ TYPE value.
- **void setBool** (bool value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- **bool getBool** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Boolean value of this Node.

- void **setByte** (unsigned char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- unsigned char **getByte** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Byte value of this Node.
- void **setChar** (char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- char **getChar** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Character value of this Node.
- void **setShort** (short value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- short **getShort** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Short value of this Node.
- void **setInt** (int value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- int **getInt** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Integer value of this Node.
- void **setLong** (long long value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- long long **getLong** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Long value of this Node.
- void **setFloat** (float value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- float **getFloat** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Float value of this Node.
- void **setDouble** (double value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- double **getDouble** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Double value of this Node.

- **void setString** (const std::string &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- **std::string getString** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the String value of this Node.
- **void setByteArray** (const std::vector< unsigned char > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- **std::vector< unsigned char > getByteArray** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Byte Array value of this Node.
- **void setList** (const decaf::util::List< PrimitiveValueNode > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- **const decaf::util::List< PrimitiveValueNode > & getList** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Primitive List value of this Node.
- **void setMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- **const decaf::util::Map< std::string, PrimitiveValueNode > & getMap** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Primitive Map value of this Node.
- **std::string toString** () const
Creates a string representation of this value.

6.525.1 Detailed Description

Class that wraps around a single value of one of the many types. Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

6.525.2 Member Enumeration Documentation

6.525.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

Enumerator:

NULL_ TYPE

BOOLEAN_TYPE
BYTE_TYPE
CHAR_TYPE
SHORT_TYPE
INTEGER_TYPE
LONG_TYPE
DOUBLE_TYPE
FLOAT_TYPE
STRING_TYPE
BYTE_ARRAY_TYPE
MAP_TYPE
LIST_TYPE
BIG_STRING_TYPE

6.525.3 Constructor & Destructor Documentation

6.525.3.1 activemq::util::PrimitiveValueNode::PrimitiveValueNode ()

Default Constructor, creates a value of the `NULL_TYPE`.

6.525.3.2 activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool *value*)

Boolean Value Constructor.

Parameters:

value - the new value to store.

6.525.3.3 activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char *value*)

Byte Value Constructor.

Parameters:

value - the new value to store.

6.525.3.4 activemq::util::PrimitiveValueNode::PrimitiveValueNode (char *value*)

Char Value Constructor.

Parameters:

value - the new value to store.

6.525.3.5 activemq::util::PrimitiveValueNode::PrimitiveValueNode (short *value*)

Short Value Constructor.

Parameters:

value - the new value to store.

6.525.3.6 activemq::util::PrimitiveValueNode::PrimitiveValueNode (int *value*)

Int Value Constructor.

Parameters:

value - the new value to store.

6.525.3.7 activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long *value*)

Long Value Constructor.

Parameters:

value - the new value to store.

6.525.3.8 activemq::util::PrimitiveValueNode::PrimitiveValueNode (float *value*)

Float Value Constructor.

Parameters:

value - the new value to store.

6.525.3.9 activemq::util::PrimitiveValueNode::PrimitiveValueNode (double *value*)

Double Value Constructor.

Parameters:

value - the new value to store.

6.525.3.10 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * *value*)

String Value Constructor.

Parameters:

value - the new value to store.

6.525.3.11 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & value)`

String Value Constructor.

Parameters:

value - the new value to store.

6.525.3.12 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::vector< unsigned char > & value)`

Byte Array Value Constructor.

Parameters:

value - the new value to store.

6.525.3.13 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::List< PrimitiveValueNode > & value)`

Primitive List Constructor.

Parameters:

value - the new value to store.

6.525.3.14 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::Map< std::string, PrimitiveValueNode > & value)`

Primitive Map Value Constructor.

Parameters:

value - the new value to store.

6.525.3.15 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const PrimitiveValueNode & node)`

Copy constructor.

Parameters:

node The instance of another node to copy to this one.

6.525.3.16 `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode () [inline]`**6.525.4** Member Function Documentation**6.525.4.1** `void activemq::util::PrimitiveValueNode::clear ()`

Clears the value from this wrapper converting it back to a blank NULL_TYPE value.

6.525.4.2 `bool activemq::util::PrimitiveValueNode::getBool () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Boolean value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.3 `unsigned char activemq::util::PrimitiveValueNode::getBytes () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Byte value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.4 `std::vector<unsigned char> activemq::util::PrimitiveValueNode::getBytesArray () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Byte Array value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.5 `char activemq::util::PrimitiveValueNode::getChar () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Character value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.6 double activemq::util::PrimitiveValueNode::getDouble () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Double value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.7 float activemq::util::PrimitiveValueNode::getFloat () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Float value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.8 int activemq::util::PrimitiveValueNode::getInt () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Integer value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.9 const decaf::util::List<PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getList () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Primitive List value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.10 `long long activemq::util::PrimitiveValueNode::getLong () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Long value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.11 `const decaf::util::Map<std::string, PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getMap () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Primitive Map value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.12 `short activemq::util::PrimitiveValueNode::getShort () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Short value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.13 `std::string activemq::util::PrimitiveValueNode::getString () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the String value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.14 PrimitiveType activemq::util::PrimitiveValueNode::getType () const
[inline]

Gets the Value Type of this type wrapper.

Returns:

the PrimitiveType value for this wrapper.

6.525.4.15 PrimitiveValue activemq::util::PrimitiveValueNode::getValue () const
[inline]

Gets the internal Primitive Value object from this wrapper.

Returns:

a copy of the contained **PrimitiveValue** (p. 2551)

6.525.4.16 PrimitiveValueNode& activemq::util::PrimitiveValueNode::operator=
(const PrimitiveValueNode & *node*)

Assignment operator, copies the data from the other node.

Parameters:

node The instance of another node to copy to this one.

6.525.4.17 bool activemq::util::PrimitiveValueNode::operator== (const
PrimitiveValueNode & *node*) const

Comparison Operator, compares this node to the other node.

Returns:

true if the values are the same false otherwise.

6.525.4.18 void activemq::util::PrimitiveValueNode::setBool (bool *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.19 void activemq::util::PrimitiveValueNode::setByte (unsigned char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.20 void activemq::util::PrimitiveValueNode::setByteArray (const std::vector< unsigned char > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.21 void activemq::util::PrimitiveValueNode::setChar (char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.22 void activemq::util::PrimitiveValueNode::setDouble (double *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.23 void activemq::util::PrimitiveValueNode::setFloat (float *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.24 void activemq::util::PrimitiveValueNode::setInt (int *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.25 void activemq::util::PrimitiveValueNode::setList (const decaf::util::List< PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.26 void activemq::util::PrimitiveValueNode::setLong (long long *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.27 void activemq::util::PrimitiveValueNode::setMap (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.28 void activemq::util::PrimitiveValueNode::setShort (short *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.29 void activemq::util::PrimitiveValueNode::setString (const std::string & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.525.4.30 void activemq::util::PrimitiveValueNode::setValue (const PrimitiveValue & *value*, PrimitiveType *valueType*)

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

Parameters:

value The value to set as the value contained in this Node.

valueType The type of the value being set into this one.

6.525.4.31 `std::string activemq::util::PrimitiveValueNode::toString () const`

Creates a string representation of this value.

Returns:

string value of this type wrapper.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

6.526 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

#include <src/main/decaf/security/Principal.h> Inheritance diagram for decaf::security::Principal:

Public Member Functions

- virtual **~Principal** ()
- virtual bool **equals** (const **Principal** &another) const =0
Compares two principals to see if they are the same.
- virtual std::string **getName** () const =0
Provides the name of this principal.

6.526.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

6.526.2 Constructor & Destructor Documentation

6.526.2.1 virtual decaf::security::Principal::~~Principal () [inline, virtual]

6.526.3 Member Function Documentation

6.526.3.1 virtual bool decaf::security::Principal::equals (const **Principal** & *another*) const [pure virtual]

Compares two principals to see if they are the same.

Parameters:

another A principal to be tested for equality to this one.

Returns:

true if the given principal is equivalent to this one.

6.526.3.2 virtual std::string decaf::security::Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns:

the name of this principal.

Implemented in **decaf::security::auth::x500::X500Principal** (p. 3406).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Principal.h`

6.527 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

#include <src/main/decaf/util/PriorityQueue.h> Inheritance diagram for decaf::util::PriorityQueue< E >:

Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

Public Member Functions

- **PriorityQueue** ()
*Creates a **Priority Queue** (p. 2671) with the default initial capacity.*
- **PriorityQueue** (std::size_t initialCapacity)
*Creates a **Priority Queue** (p. 2671) with the capacity value supplied.*
- **PriorityQueue** (std::size_t initialCapacity, **Comparator**< E > *comparator)
*Creates a **Priority Queue** (p. 2671) with the default initial capacity.*
- **PriorityQueue** (const **Collection**< E > &source)
*Creates a **PriorityQueue** (p. 2571) containing the elements in the specified **Collection** (p. 1054).*
- **PriorityQueue** (const **PriorityQueue**< E > &source)
*Creates a **PriorityQueue** (p. 2571) containing the elements in the specified priority queue.*
- virtual ~**PriorityQueue** ()
- **PriorityQueue**< E > & **operator=** (const **Collection**< E > &source)
*Assignment operator, assign another **Collection** (p. 1054) to this one.*
- **PriorityQueue**< E > & **operator=** (const **PriorityQueue**< E > &source)
*Assignment operator, assign another **PriorityQueue** (p. 2571) to this one.*
- virtual **decaf::util::Iterator**< E > * **iterator** ()
- virtual **decaf::util::Iterator**< E > * **iterator** () const
- virtual std::size_t **size** () const
Returns the number of elements in this collection.
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all elements of the queue.
- virtual bool **offer** (const E &value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into the queue provided that the condition allows such an operation.

- virtual bool **poll** (E &result)

Gets and removes the element in the head of the queue.

- virtual bool **peek** (E &result) const

Gets but not removes the element in the head of the queue.

- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)

Retrieves and removes the head of this queue.

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an *IllegalStateException* if no space is currently available.*

- decaf::lang::Pointer< **Comparator**< E > > **comparator** () const

*obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 2571) is using to compare the elements in the queue with.*

Friends

- class **PriorityQueueIterator**

6.527.1 Detailed Description

template<typename E> class decaf::util::PriorityQueue< E >

An unbounded priority queue based on a binary heap algorithm. The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p. 1086) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p. 1054) and **Iterator** (p. 1832) interfaces. The **Iterator** (p. 1832) provided in method **iterator()** (p. 2575) is

not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort(pq.toArray())`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p. 2571) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe `PriorityBlockingQueue` class.

Implementation note: this implementation provides $O(\log(n))$ time for the enqueueing and dequeuing methods (`offer`, `poll`, **`remove()`** (p. 2577) and `add`); linear time for the `remove(Object)` and `contains(Object)` methods; and constant time for the retrieval methods (`peek`, `element`, and `size`).

Since:

1.0

6.527.2 Constructor & Destructor Documentation

6.527.2.1 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue()` [inline]

Creates a **Priority Queue** (p. 2671) with the default initial capacity.

6.527.2.2 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue(std::size_t initialCapacity)` [inline]

Creates a **Priority Queue** (p. 2671) with the capacity value supplied.

Parameters:

initialCapacity The initial number of elements allocated to this **PriorityQueue** (p. 2571).

6.527.2.3 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue(std::size_t initialCapacity, Comparator< E > * comparator)` [inline]

Creates a **Priority Queue** (p. 2671) with the default initial capacity. This new **PriorityQueue** (p. 2571) takes ownership of the passed **Comparator** (p. 1086) instance and uses that to determine the ordering of the elements in the **Queue** (p. 2671).

Parameters:

initialCapacity The initial number of elements allocated to this **PriorityQueue** (p. 2571).

comparator The **Comparator** (p. 1086) instance to use in sorting the elements in the **Queue** (p. 2671).

Exceptions:

NullPointerException if the passed **Comparator** (p. 1086) is NULL.

6.527.2.4 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue(const Collection< E > & source)` [inline]

Creates a **PriorityQueue** (p. 2571) containing the elements in the specified **Collection** (p. 1054).

Parameters:

source the **Collection** (p. 1054) whose elements are to be placed into this priority queue

6.527.2.5 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue
(const PriorityQueue< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2571) containing the elements in the specified priority queue. This priority queue will be ordered according to the same ordering as the given priority queue.

Parameters:

source the priority queue whose elements are to be placed into this priority queue

6.527.2.6 `template<typename E> virtual decaf::util::PriorityQueue< E
>::~PriorityQueue () [inline, virtual]`

6.527.3 Member Function Documentation

6.527.3.1 `template<typename E> virtual bool decaf::util::PriorityQueue<
E >::add (const E & value) throw (
lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException,
lang::exceptions::IllegalStateException) [inline,
virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available. This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

Parameters:

value - the element to offer to the **Queue** (p. 2671).

Returns:

true if the add succeeds.

Exceptions:

IllegalArgumentException if the element cannot be added.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 163).

References `DECAF_CATCH_EXCEPTION_CONVERT`, `DECAF_CATCH_RETHROW`, `DECAF_CATCHALL_THROW`, and `decaf::util::PriorityQueue< E >::offer()`.

6.527.3.2 `template<typename E> virtual void decaf::util::PriorityQueue< E
>::clear () throw (lang::exceptions::UnsupportedOperationException)
[inline, virtual]`

Removes all elements of the queue. This implementation repeatedly invokes `poll` until it returns the empty marker.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 164).

6.527.3.3 `template<typename E> decaf::lang::Pointer< Comparator<E> > decaf::util::PriorityQueue< E >::comparator () const [inline]`

obtains a Copy of the Pointer instance that this **PriorityQueue** (p.2571) is using to compare the elements in the queue with. The returned value is a copy, the caller cannot change the value if the internal Pointer value.

Returns:

a copy of the **Comparator** (p.1086) Pointer being used by this **Queue** (p.2671).

6.527.3.4 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator () const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p.1830).

6.527.3.5 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator () [inline, virtual]`

Returns:

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p.1830).

References **decaf::util::PriorityQueue< E >::PriorityQueueIterator**.

6.527.3.6 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (const E & value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation. The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters:

value the specified element to insert into the queue.

Returns:

true if the operation succeeds and false if it fails.

Exceptions:

NullPointerException if the **Queue** (p.2671) implementation does not allow Null values to be inserted into the **Queue** (p.2671).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p.2672).

Referenced by **decaf::util::PriorityQueue< E >::add()**.

6.527.3.7 `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator= (const PriorityQueue< E > & source) [inline]`

Assignment operator, assign another **PriorityQueue** (p. 2571) to this one.

Parameters:

source The **PriorityQueue** (p. 2571) to copy to this one.

6.527.3.8 `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator= (const Collection< E > & source) [inline]`

Assignment operator, assign another **Collection** (p. 1054) to this one.

Parameters:

source The **Collection** (p. 1054) to copy to this one.

6.527.3.9 `template<typename E> virtual bool decaf::util::PriorityQueue< E>::peek (E & result) const [inline, virtual]`

Gets but not removes the element in the head of the queue. The result if successful is assigned to the result parameter.

Parameters:

result Reference to an instance of the contained type to assigned the removed value to.

Returns:

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2672).

6.527.3.10 `template<typename E> virtual bool decaf::util::PriorityQueue< E>::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue. If the operation succeeds the value of the element at the head of the **Queue** (p. 2671) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters:

result Reference to an instance of the contained type to assigned the removed value to.

Returns:

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2673).

6.527.3.11 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation). More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p.1054).

Reimplemented from `decaf::util::AbstractCollection< E >` (p.155).

6.527.3.12 `template<typename E> virtual E decaf::util::PriorityQueue< E >::remove () throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Retrieves and removes the head of this queue. This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

Returns:

a copy of the element in the head of the queue.

Exceptions:

NoSuchElementException if the queue is empty.

Reimplemented from `decaf::util::AbstractQueue< E >` (p.164).

6.527.3.13 `template<typename E> virtual std::size_t decaf::util::PriorityQueue< E >::size () const [inline, virtual]`

Returns the number of elements in this collection. If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

Returns:

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 1062).

6.527.4 Friends And Related Function Documentation

6.527.4.1 `template<typename E> friend class PriorityQueueIterator` [friend]

Referenced by `decaf::util::PriorityQueue`< **E** >::iterator().

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ PriorityQueue.h`

6.528 activemq::commands::ProducerAck Class Reference

#include <src/main/activemq/commands/ProducerAck.h> Inheritance diagram for activemq::commands::ProducerAck:

Public Member Functions

- **ProducerAck** ()
- virtual **~ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual int **getSize** () const
- virtual void **setSize** (int size)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERACK** = 19

Protected Member Functions

- **ProducerAck** (const **ProducerAck** &)
- **ProducerAck** & **operator=** (const **ProducerAck** &)

Protected Attributes

- `Pointer< ProducerId > producerId`
- `int size`

6.528.1 Constructor & Destructor Documentation

6.528.1.1 `activemq::commands::ProducerAck::ProducerAck (const ProducerAck &) [inline, protected]`

6.528.1.2 `activemq::commands::ProducerAck::ProducerAck ()`

6.528.1.3 `virtual activemq::commands::ProducerAck::~~ProducerAck () [virtual]`

6.528.2 Member Function Documentation

6.528.2.1 `virtual ProducerAck* activemq::commands::ProducerAck::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.528.2.2 `virtual void activemq::commands::ProducerAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.528.2.3 `virtual bool activemq::commands::ProducerAck::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.528.2.4 virtual unsigned char activemq::commands::ProducerAck::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.528.2.5 virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () [virtual]

6.528.2.6 virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () const [virtual]

6.528.2.7 virtual int activemq::commands::ProducerAck::getSize () const [virtual]

6.528.2.8 virtual bool activemq::commands::ProducerAck::isProducerAck () const [inline, virtual]

Returns:

an answer of true to the **isProducerAck()** (p. 2581) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 654).

6.528.2.9 ProducerAck& activemq::commands::ProducerAck::operator= (const ProducerAck &) [inline, protected]

6.528.2.10 virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]

6.528.2.11 virtual void activemq::commands::ProducerAck::setSize (int *size*) [virtual]

6.528.2.12 virtual std::string activemq::commands::ProducerAck::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.528.2.13 `virtual Pointer<Command> activemq::commands::ProducerAck::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.528.3 Field Documentation

6.528.3.1 `const unsigned char activemq::commands::ProducerAck::ID_-
PRODUCERACK = 19 [static]`

6.528.3.2 `Pointer<ProducerId> activemq::commands::ProducerAck::producerId
[protected]`

6.528.3.3 `int activemq::commands::ProducerAck::size [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerAck.h`

6.529 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2583).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.529.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2583). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.529.2 Constructor & Destructor Documentation

6.529.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

6.529.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

6.529.3 Member Function Documentation

6.529.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.529.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.529.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 686).

6.529.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 687).

6.529.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 688).

6.529.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.529.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerAckMarshaller.h**

6.530 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2587).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.530.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2587). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.530.2 Constructor & Destructor Documentation

6.530.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

6.530.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

6.530.3 Member Function Documentation

6.530.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.530.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.530.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.530.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.530.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.530.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 661).

6.530.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 662).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h`

6.531 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2591).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.531.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2591). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.531.2 Constructor & Destructor Documentation

6.531.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

6.531.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

6.531.3 Member Function Documentation

6.531.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.531.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.531.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 672).

6.531.3.4 **virtual void ac-**
tivemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseUnmarshal
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 673).

6.531.3.5 **virtual int ac-**
tivemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal1
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure** *
dataStructure, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 674).

6.531.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.531.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerAckMarshaller.h**

6.532 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2595).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.532.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2595). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.532.2 Constructor & Destructor Documentation

6.532.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

6.532.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

6.532.3 Member Function Documentation

6.532.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.532.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.532.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 679).

6.532.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 680).

6.532.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 681).

6.532.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.532.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerAckMarshaller.h**

6.533 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2599).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.533.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2599). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.533.2 Constructor & Destructor Documentation

6.533.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarshaller()` [inline]

6.533.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~~ProducerAckMarshaller()` [inline, virtual]

6.533.3 Member Function Documentation

6.533.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.533.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.533.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.533.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.533.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.533.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.533.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**

6.534 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

#include <src/main/activemq/cmsutil/ProducerCallback.h> Inheritance diagram for activemq::cmsutil::ProducerCallback:

Public Member Functions

- virtual **~ProducerCallback** ()
- virtual void **doInCms** (cms::Session *session, cms::MessageProducer *producer)=0
throw (cms::CMSException)

Execute an action given a session and producer.

6.534.1 Detailed Description

Callback for sending a message to a CMS destination.

6.534.2 Constructor & Destructor Documentation

- 6.534.2.1** virtual activemq::cmsutil::ProducerCallback::~~ProducerCallback ()
[inline, virtual]

6.534.3 Member Function Documentation

- 6.534.3.1** virtual void activemq::cmsutil::ProducerCallback::doInCms
(cms::Session * *session*, cms::MessageProducer * *producer*) throw (cms::CMSException) [pure virtual]

Execute an action given a session and producer.

Parameters:

session the CMS Session

producer the CMS Producer

Exceptions:

cms::CMSException (p. 1031) if thrown by CMS API methods

Implemented in **activemq::cmsutil::CmsTemplate::SendExecutor** (p. 2836).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/ProducerCallback.h

6.535 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ProducerExecutor:

Public Member Functions

- **ProducerExecutor** (**ProducerCallback** **action*, **CmsTemplate** **parent*, **cms::Destination** **destination*)
- virtual **~ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** **session*) throw (**cms::CMSEException**)
Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** * **getDestination** (**cms::Session** **session* **AMQCPP_UNUSED**) throw (**cms::CMSEException**)

Protected Attributes

- **ProducerCallback** * *action*
- **CmsTemplate** * *parent*
- **cms::Destination** * *destination*

6.535.1 Constructor & Destructor Documentation

- 6.535.1.1** **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** (**ProducerCallback** * *action*, **CmsTemplate** * *parent*, **cms::Destination** * *destination*) [inline]
- 6.535.1.2** virtual **activemq::cmsutil::CmsTemplate::ProducerExecutor::~~ProducerExecutor** () [inline, virtual]

6.535.2 Member Function Documentation

- 6.535.2.1** virtual void **activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms** (**cms::Session** * *session*) throw (**cms::CMSEException**) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters:

session the CMS Session

Exceptions:

cms::CMSEException (p. 1031) if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 2852).

6.535.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]

6.535.3 Field Documentation

6.535.3.1 ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]

6.535.3.2 cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]

6.535.3.3 CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.536 activemq::commands::ProducerId Class Reference

#include <src/main/activemq/commands/ProducerId.h> Inheritance diagram for activemq::commands::ProducerId:

Public Types

- typedef decaf::lang::PointerComparator< **ProducerId** > **COMPARATOR**

Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual int **compareTo** (const **ProducerId** &value) const
- virtual bool **equals** (const **ProducerId** &value) const
- virtual bool **operator==** (const **ProducerId** &value) const
- virtual bool **operator<** (const **ProducerId** &value) const
- **ProducerId** & **operator=** (const **ProducerId** &other)

Static Public Attributes

- static const unsigned char **ID_PRODUCERID** = 123

Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

6.536.1 Member Typedef Documentation

6.536.1.1 `typedef decaf::lang::PointerComparator<ProducerId>
activemq::commands::ProducerId::COMPARATOR`

6.536.2 Constructor & Destructor Documentation

6.536.2.1 `activemq::commands::ProducerId::ProducerId ()`

6.536.2.2 `activemq::commands::ProducerId::ProducerId (const ProducerId &
other)`

6.536.2.3 `activemq::commands::ProducerId::ProducerId (const SessionId &
sessionId, long long consumerId) [inline]`

References `activemq::commands::SessionId::getConnectionId()`, and `activemq::commands::SessionId::getValue()`.

6.536.2.4 `virtual activemq::commands::ProducerId::~~ProducerId () [virtual]`

6.536.3 Member Function Documentation

6.536.3.1 `virtual ProducerId* ac-
tivemq::commands::ProducerId::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.536.3.2 `virtual int activemq::commands::ProducerId::compareTo (const
ProducerId & value) const [virtual]`

6.536.3.3 `virtual void activemq::commands::ProducerId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.536.3.4 `virtual bool activemq::commands::ProducerId::equals (const ProducerId & value) const` [virtual]

6.536.3.5 `virtual bool activemq::commands::ProducerId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.536.3.6 `virtual std::string& activemq::commands::ProducerId::getConnectionId ()` [virtual]

6.536.3.7 `virtual const std::string& activemq::commands::ProducerId::getConnectionId () const` [virtual]

6.536.3.8 `virtual unsigned char activemq::commands::ProducerId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.536.3.9 `const Pointer<SessionId>& activemq::commands::ProducerId::getParentId () const`
- 6.536.3.10 `virtual long long activemq::commands::ProducerId::getSessionId () const [virtual]`
- 6.536.3.11 `virtual long long activemq::commands::ProducerId::getValue () const [virtual]`
- 6.536.3.12 `virtual bool activemq::commands::ProducerId::operator< (const ProducerId & value) const [virtual]`
- 6.536.3.13 `ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & other)`
- 6.536.3.14 `virtual bool activemq::commands::ProducerId::operator== (const ProducerId & value) const [virtual]`
- 6.536.3.15 `virtual void activemq::commands::ProducerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.536.3.16 `virtual void activemq::commands::ProducerId::setSessionId (long long sessionId) [virtual]`
- 6.536.3.17 `virtual void activemq::commands::ProducerId::setValue (long long value) [virtual]`
- 6.536.3.18 `virtual std::string activemq::commands::ProducerId::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p.718).

6.536.4 Field Documentation

- 6.536.4.1 `std::string activemq::commands::ProducerId::connectionId [protected]`
- 6.536.4.2 `const unsigned char activemq::commands::ProducerId::ID_ - PRODUCERID = 123 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.536.4.3 `long long activemq::commands::ProducerId::sessionId` [protected]

6.536.4.4 `long long activemq::commands::ProducerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerId.h`

6.537 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2611).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.537.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2611). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.537.2 Constructor & Destructor Documentation

6.537.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.537.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.537.3 Member Function Documentation

6.537.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.537.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.537.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.537.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.537.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.537.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.537.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerIdMarshaller.h**

6.538 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2615).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.538.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2615). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.538.2 Constructor & Destructor Documentation

6.538.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.538.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.538.3 Member Function Documentation

6.538.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.538.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.538.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.538.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.538.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.538.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.538.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerIdMarshaller.h**

6.539 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2619).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.539.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2619). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.539.2 Constructor & Destructor Documentation

6.539.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.539.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.539.3 Member Function Documentation

6.539.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.539.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.539.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.539.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.539.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.539.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.539.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerIdMarshaller.h**

6.540 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2623).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.540.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2623). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.540.2 Constructor & Destructor Documentation

6.540.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.540.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.540.3 Member Function Documentation

6.540.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.540.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.540.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.540.3.4 virtual void **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.540.3.5 virtual int **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.540.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.540.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerIdMarshaller.h**

6.541 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2627).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.541.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2627). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.541.2 Constructor & Destructor Documentation

6.541.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.541.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.541.3 Member Function Documentation

6.541.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.541.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.541.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.541.3.4 virtual void **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.541.3.5 virtual int **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.541.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.541.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerIdMarshaller.h**

6.542 activemq::commands::ProducerInfo Class Reference

#include <src/main/activemq/commands/ProducerInfo.h> Inheritance diagram for activemq::commands::ProducerInfo:

Public Member Functions

- **ProducerInfo** ()
- virtual **~ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int windowSize)
- virtual bool **isProducerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERINFO** = 6

Protected Member Functions

- **ProducerInfo** (const **ProducerInfo** &)
- **ProducerInfo** & **operator=** (const **ProducerInfo** &)

Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::vector< decaf::lang::Pointer< **BrokerId** > > **brokerPath**
- bool **dispatchAsync**
- int **windowSize**

6.542.1 Constructor & Destructor Documentation

6.542.1.1 **activemq::commands::ProducerInfo::ProducerInfo** (const **ProducerInfo** &) [inline, protected]

6.542.1.2 **activemq::commands::ProducerInfo::ProducerInfo** ()

6.542.1.3 **virtual** **activemq::commands::ProducerInfo::~~ProducerInfo** () [virtual]

6.542.2 Member Function Documentation

6.542.2.1 **virtual** **ProducerInfo*** **activemq::commands::ProducerInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

6.542.2.2 **virtual** void **activemq::commands::ProducerInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.542.2.3 virtual bool activemq::commands::ProducerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.542.2.4 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () [virtual]

6.542.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () const [virtual]

6.542.2.6 virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.542.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination ()`
[virtual]
- 6.542.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () const` [virtual]
- 6.542.2.9 `virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId ()`
[virtual]
- 6.542.2.10 `virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () const`
[virtual]
- 6.542.2.11 `virtual int activemq::commands::ProducerInfo::getWindowSize () const`
[virtual]
- 6.542.2.12 `virtual bool activemq::commands::ProducerInfo::isDispatchAsync () const` [virtual]
- 6.542.2.13 `virtual bool activemq::commands::ProducerInfo::isProducerInfo () const`
[inline, virtual]

Returns:

an answer of true to the `isProducerInfo()` (p. 2634) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 654).

- 6.542.2.14 `ProducerInfo& activemq::commands::ProducerInfo::operator= (const ProducerInfo &)` [inline, protected]
- 6.542.2.15 `virtual void activemq::commands::ProducerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)`
[virtual]
- 6.542.2.16 `virtual void activemq::commands::ProducerInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.542.2.17 `virtual void activemq::commands::ProducerInfo::setDispatchAsync (bool dispatchAsync)` [virtual]
- 6.542.2.18 `virtual void activemq::commands::ProducerInfo::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.542.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize (int windowSize)` [virtual]
- 6.542.2.20 `virtual std::string activemq::commands::ProducerInfo::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

6.542.2.21 `virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.542.3 Field Documentation

6.542.3.1 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ProducerInfo::brokerPath` [protected]

6.542.3.2 `Pointer<ActiveMQDestination> activemq::commands::ProducerInfo::destination` [protected]

6.542.3.3 `bool activemq::commands::ProducerInfo::dispatchAsync` [protected]

6.542.3.4 `const unsigned char activemq::commands::ProducerInfo::ID_PRODUCERINFO = 6` [static]

6.542.3.5 `Pointer<ProducerId> activemq::commands::ProducerInfo::producerId` [protected]

6.542.3.6 `int activemq::commands::ProducerInfo::windowSize` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerInfo.h`

6.543 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2636).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.543.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2636). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.543.2 Constructor & Destructor Documentation

6.543.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

6.543.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

6.543.3 Member Function Documentation

6.543.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.543.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.543.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.543.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.543.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.543.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.543.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerInfoMarshaller.h**

6.544 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2640).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.544.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2640). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.544.2 Constructor & Destructor Documentation

6.544.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::ProducerInfoMarshaller()` `[inline]`

6.544.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` `[inline, virtual]`

6.544.3 Member Function Documentation

6.544.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::createObject() const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.544.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::getDataStructureType() const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.544.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.544.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.544.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.544.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.544.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerInfoMarshaller.h**

6.545 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2644).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.545.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2644). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.545.2 Constructor & Destructor Documentation

6.545.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::ProducerInfoMarshaller()` `[inline]`

6.545.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` `[inline, virtual]`

6.545.3 Member Function Documentation

6.545.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.545.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.545.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.545.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.545.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.545.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.545.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerInfoMarshaller.h**

6.546 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2648).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.546.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2648). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.546.2 Constructor & Destructor Documentation

6.546.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

6.546.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

6.546.3 Member Function Documentation

6.546.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.546.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.546.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.546.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.546.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.546.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.546.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**

6.547 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2652).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.547.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2652). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.547.2 Constructor & Destructor Documentation

6.547.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarshaller()` [inline]

6.547.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~~ProducerInfoMarshaller()` [inline, virtual]

6.547.3 Member Function Documentation

6.547.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.547.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.547.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.547.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.547.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.547.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.547.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**

6.548 activemq::state::ProducerState Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual **~ProducerState** ()
- std::string **toString** () const
- const **Pointer**< **ProducerInfo** > & **getInfo** () const

6.548.1 Constructor & Destructor Documentation

6.548.1.1 **activemq::state::ProducerState::ProducerState** (const **Pointer**< **ProducerInfo** > & *info*)

6.548.1.2 virtual **activemq::state::ProducerState::~~ProducerState** () [virtual]

6.548.2 Member Function Documentation

6.548.2.1 const **Pointer**<**ProducerInfo**>& **activemq::state::ProducerState::getInfo** () const [inline]

6.548.2.2 std::string **activemq::state::ProducerState::toString** () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ProducerState.h**

6.549 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator=** (const **Properties** &src)
Assignment Operator.
- bool **isEmpty** () const
Returns true if the properties object is empty.
- std::size_t **size** () const
- const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- void **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- void **remove** (const std::string &name)
Removes the property with the given name.
- std::vector< std::pair< std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an array.
- void **copy** (const **Properties** &source)
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 2657) instance in NULL then this **List** (p. 1984) is not modified.*
- **Properties** * **clone** () const
Clones this object.
- void **clear** ()
Clears all properties from the map.
- bool **equals** (const **Properties** &source) const
*Test whether two **Properties** (p. 2657) objects are equivalent.*
- std::string **toString** () const

*Formats the contents of the **Properties** (p. 2657) Object into a string that can be logged, etc.*

- void **load** (decaf::io::InputStream *stream) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input byte stream.

- void **load** (decaf::io::Reader *reader) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

- void **store** (decaf::io::OutputStream *out, const std::string &comment) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

*Writes this property list (key and element pairs) in this **Properties** (p. 2657) table to the output stream in a format suitable for loading into a **Properties** (p. 2657) table using the load(InputStream) method.*

- void **store** (decaf::io::Writer *writer, const std::string &comments) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

*Writes this property list (key and element pairs) in this **Properties** (p. 2657) table to the output character stream in a format that can be read by the load(Reader) method.*

Protected Attributes

- std::auto_ptr< **Properties** > **defaults**

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

6.549.1 Detailed Description

Java-like properties class for mapping string names to string values. The **Properties** (p. 2657) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 2657) instance can contain an internal **Properties** (p. 2657) list that contains default values for keys not found in the **Properties** (p. 2657) **List** (p. 1984).

The **Properties** (p. 2657) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since:

1.0

6.549.2 Constructor & Destructor Documentation

6.549.2.1 `decaf::util::Properties::Properties ()`

6.549.2.2 `decaf::util::Properties::Properties (const Properties & src)`

6.549.2.3 `virtual decaf::util::Properties::~~Properties ()` [virtual]

6.549.3 Member Function Documentation

6.549.3.1 `void decaf::util::Properties::clear ()`

Clears all properties from the map.

6.549.3.2 `Properties* decaf::util::Properties::clone () const`

Clones this object.

Returns:

a replica of this object.

6.549.3.3 `void decaf::util::Properties::copy (const Properties & source)`

Copies the contents of the given properties object to this one, if the given **Properties** (p. 2657) instance is NULL then this **List** (p. 1984) is not modified.

Parameters:

source The source properties object.

6.549.3.4 `bool decaf::util::Properties::equals (const Properties & source) const`

Test whether two **Properties** (p. 2657) objects are equivalent. Two **Properties** (p. 2657) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

Parameters:

source The **Properties** (p. 2657) object to compare this instance to.

Returns:

true if the contents of the two **Properties** (p. 2657) objects are the same.

6.549.3.5 `std::string decaf::util::Properties::getProperty (const std::string & name, const std::string & default Value) const`

Looks up the value for the given property.

Parameters:

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns:

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

6.549.3.6 `const char* decaf::util::Properties::getProperty (const std::string & name) const`

Looks up the value for the given property.

Parameters:

name The name of the property to be looked up.

Returns:

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Referenced by `decaf::util::logging::LogManager::getProperty()`.

6.549.3.7 `bool decaf::util::Properties::hasProperty (const std::string & name) const`

Check to see if the Property exists in the set.

Parameters:

name - property name to check for in this properties set.

Returns:

true if property exists, false otherwise.

6.549.3.8 `bool decaf::util::Properties::isEmpty () const`

Returns true if the properties object is empty.

Returns:

true if empty

6.549.3.9 `void decaf::util::Properties::load (decaf::io::Reader * reader) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)`

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format. **Properties** (p. 2657) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character `\`. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII `'#'` or `'!'` as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (`' '`), tab (`'\t'`), and form feed (`'\f'`) to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of $2n$ contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped `'='`, `':'`, or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\: \=
```

would be the two-character key `":="`. Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is `'='` or `':'`, then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string `""`. Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key `"Truth"` and the associated element value `"Beauty"`:

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is `"fruits"` and the associated element is: `"apple, banana, pear, cantaloupe, watermelon, kiwi, mango"`

Note that a space appears before each `\` so that a space will appear after each comma in the final result; the `\`, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is `"cheeses"` and the associated element is the empty string `""`.

Characters in keys and elements can be represented in escape sequences similar to those used

for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.
- The method does not treat a backslash character, `\`, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence `"\z"` would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence `"\b"` as equivalent to the single character `'b'`.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

Parameters:

reader The Reader that provides an character stream as input.

Exceptions:

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

6.549.3.10 `void decaf::util::Properties::load (decaf::io::InputStream
* stream) throw (decaf::io::IOException, de-
caf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::NullPointerException)`

Reads a property list (key and element pairs) from the input byte stream. The input stream is in a simple line-oriented format as specified in `load(Reader)` and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

Parameters:

stream The stream to read the properties data from.

Exceptions:

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

6.549.3.11 Properties& decaf::util::Properties::operator= (const Properties & *src*)

Assignment Operator.

Parameters:

src The **Properties** (p. 2657) list to copy to this **List** (p. 1984).

Returns:

a reference to this **List** (p. 1984) for use in chaining.

6.549.3.12 void decaf::util::Properties::remove (const std::string & *name*)

Removes the property with the given name.

Parameters:

name the name of the property to remove.

**6.549.3.13 void decaf::util::Properties::setProperty (const std::string & *name*,
const std::string & *value*)**

Sets the value for a given property. If the property already exists, overwrites the value.

Parameters:

name The name of the value to be written.

value The value to be written.

6.549.3.14 std::size_t decaf::util::Properties::size () const**Returns:**

The number of **Properties** (p. 2657) in this **Properties** (p. 2657) Object.

**6.549.3.15 void decaf::util::Properties::store (decaf::io::Writer * *writer*,
const std::string & *comments*) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)**

Writes this property list (key and element pairs) in this **Properties** (p. 2657) table to the output character stream in a format that can be read by the load(Reader) method. **Properties** (p. 2657) from the defaults table of this **Properties** (p. 2657) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (

'), a carriage return ("), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII `#` character, the current date and time (as if produced by the `toString` method of **Date** (p.1467) for the current time), and a line separator as generated by the **Writer**.

Then every entry in this **Properties** (p.2657) table is written out, one per line. For each entry the key string is written, then an ASCII `=`, then the associated element string. For the key, all space characters are written with a preceding `\` character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding `\` character. The key and element characters `#`, `!`, `=`, and `:` are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters:

- writer* The **Writer** instance to use to output the properties.
- comments* A description of these properties that is written before writing the properties.

Exceptions:

- IOException* if there is an error while writing from the stream.
- NullPointerException* if the passed stream is **Null**.

6.549.3.16 `void decaf::util::Properties::store (decaf::io::OutputStream * out,
const std::string & comment) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)`

Writes this property list (key and element pairs) in this **Properties** (p.2657) table to the output stream in a format suitable for loading into a **Properties** (p.2657) table using the `load(InputStream)` method. **Properties** (p.2657) from the defaults table of this **Properties** (p.2657) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in `store(Writer)`, with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value `xxxx`.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value `xxxx`.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters:

- out* The **OutputStream** instance to write the properties to.
- comment* A description of these properties that is written to the output stream.

Exceptions:

- IOException* if there is an error while writing from the stream.
- NullPointerException* if the passed stream is **Null**.

6.549.3.17 `std::vector< std::pair< std::string, std::string > > decaf::util::Properties::toArray () const`

Method that serializes the contents of the property map to an array.

Returns:

list of pairs where the first is the name and the second is the value.

6.549.3.18 `std::string decaf::util::Properties::toString () const`

Formats the contents of the **Properties** (p. 2657) Object into a string that can be logged, etc.

Returns:

string value of this object.

6.549.4 Field Documentation

6.549.4.1 `std::auto_ptr<Properties> decaf::util::Properties::defaults` [protected]

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

6.550 decaf::util::logging::PropertiesChangeListener Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 2657).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

Public Member Functions

- virtual **~PropertiesChangeListener** ()
- virtual void **onPropertyChanged** (const std::string &name, const std::string &oldValue, const std::string &newValue)=0

Change Event, called when a property is changed.

6.550.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 2657).

6.550.2 Constructor & Destructor Documentation

- 6.550.2.1 virtual
decaf::util::logging::PropertiesChangeListener::~~PropertiesChangeListener
() [inline, virtual]

6.550.3 Member Function Documentation

- 6.550.3.1 virtual void de-
caf::util::logging::PropertiesChangeListener::onPropertyChanged (const
std::string & *name*, const std::string & *oldValue*, const std::string &
newValue) [pure virtual]

Change Event, called when a property is changed.

Parameters:

name - Name of the Property

oldValue - Old Value of the Property

newValue - New Value of the Property

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**

6.551 decaf::net::ProtocolException Class Reference

#include <src/main/decaf/net/ProtocolException.h> Inheritance diagram for decaf::net::ProtocolException:

Public Member Functions

- **ProtocolException** () throw ()
Default Constructor.
- **ProtocolException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **ProtocolException** (const **ProtocolException** &ex) throw ()
Copy Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ProtocolException** (const std::exception *cause) throw ()
Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ProtocolException** * clone () const
Clones this exception.
- virtual ~**ProtocolException** () throw ()

6.551.1 Constructor & Destructor Documentation

6.551.1.1 decaf::net::ProtocolException::ProtocolException () throw () [inline]

Default Constructor.

6.551.1.2 decaf::net::ProtocolException::ProtocolException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.551.1.3 `decaf::net::ProtocolException::ProtocolException (const ProtocolException & ex) throw ()` [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.551.1.4 `decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.551.1.5 `decaf::net::ProtocolException::ProtocolException (const std::exception * cause) throw ()` [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.551.1.6 `decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.551.1.7 `virtual decaf::net::ProtocolException::~~ProtocolException () throw ()`
 [inline, virtual]

6.551.2 Member Function Documentation

6.551.2.1 `virtual ProtocolException* decaf::net::ProtocolException::clone () const`
 [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ProtocolException.h`

6.552 decaf::security::PublicKey Class Reference

A public key.

`#include <src/main/decaf/security/PublicKey.h>`
Inheritance diagram for decaf::security::PublicKey:

Public Member Functions

- virtual `~PublicKey ()`

6.552.1 Detailed Description

A public key. This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

6.552.2 Constructor & Destructor Documentation

6.552.2.1 virtual `decaf::security::PublicKey::~~PublicKey ()` [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/PublicKey.h`

6.553 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

#include <src/main/decaf/util/Queue.h> Inheritance diagram for decaf::util::Queue< E >:

Public Member Functions

- virtual **~Queue** ()
- virtual bool **offer** (const E &value)=0 throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into the queue provided that the condition allows such an operation.
- virtual bool **poll** (E &result)=0
Gets and removes the element in the head of the queue.
- virtual E **remove** ()=0 throw (decaf::lang::exceptions::NoSuchElementException)
Gets and removes the element in the head of the queue.
- virtual bool **peek** (E &result) const =0
Gets but not removes the element in the head of the queue.
- virtual E **element** () const =0 throw (decaf::lang::exceptions::NoSuchElementException)
Gets but not removes the element in the head of the queue.

6.553.1 Detailed Description

template<typename E> class decaf::util::Queue< E >

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection. Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

Queue (p. 2671) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Unlike the Java **Queue** (p. 2671) interface the methods of this class cannot return null to indicate that a **Queue** (p. 2671) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java **Queue** (p. 2671) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned value is to be assigned. This implies that elements in the **Queue** (p. 2671) must be *assignable* in order to utilize these methods.

Since:

1.0

6.553.2 Constructor & Destructor Documentation

6.553.2.1 `template<typename E > virtual decaf::util::Queue< E >::~~Queue ()`
`[inline, virtual]`

6.553.3 Member Function Documentation

6.553.3.1 `template<typename E > virtual E decaf::util::Queue< E >::element ()`
`const throw (decaf::lang::exceptions::NoSuchElementException) [pure`
`virtual]`

Gets but not removes the element in the head of the queue. Throws a `NoSuchElementException` if there is no element in the queue.

Returns:

the element in the head of the queue.

Exceptions:

NoSuchElementException if there is no element in the queue.

Implemented in `decaf::util::AbstractQueue< E >` (p. 164).

6.553.3.2 `template<typename E > virtual bool decaf::util::Queue< E >::offer`
`(const E & value) throw (decaf::lang::exceptions::NullPointerException,`
`decaf::lang::exceptions::IllegalArgumentException) [pure virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation. The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters:

value the specified element to insert into the queue.

Returns:

true if the operation succeeds and false if it fails.

Exceptions:

NullPointerException if the `Queue` (p. 2671) implementation does not allow Null values to be inserted into the `Queue` (p. 2671).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::PriorityQueue< E >` (p. 2575).

Referenced by `decaf::util::AbstractQueue< E >::add()`.

6.553.3.3 `template<typename E > virtual bool decaf::util::Queue< E >::peek (E &`
`result) const [pure virtual]`

Gets but not removes the element in the head of the queue. The result if successful is assigned to the result parameter.

Parameters:

result Reference to an instance of the contained type to assigned the removed value to.

Returns:

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::PriorityQueue< E >** (p. 2576).

Referenced by **decaf::util::AbstractQueue< E >::element()**.

6.553.3.4 **template<typename E > virtual bool decaf::util::Queue< E >::poll (E & result)** [pure virtual]

Gets and removes the element in the head of the queue. If the operation succeeds the value of the element at the head of the **Queue** (p. 2671) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters:

result Reference to an instance of the contained type to assigned the removed value to.

Returns:

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::PriorityQueue< E >** (p. 2576).

Referenced by **decaf::util::AbstractQueue< E >::clear()**, and **decaf::util::AbstractQueue< E >::remove()**.

6.553.3.5 **template<typename E > virtual E decaf::util::Queue< E >::remove () throw (decaf::lang::exceptions::NoSuchElementException)** [pure virtual]

Gets and removes the element in the head of the queue. Throws a **NoSuchElementException** if there is no element in the queue.

Returns:

the element in the head of the queue.

Exceptions:

NoSuchElementException if there is no element in the queue.

Implemented in **decaf::util::AbstractQueue< E >** (p. 164), and **decaf::util::PriorityQueue< E >** (p. 2577).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Queue.h`

6.554 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

#include <src/main/cms/Queue.h> Inheritance diagram for cms::Queue:

Public Member Functions

- virtual `~Queue ()`
- virtual `std::string getQueueName () const =0 throw (CMSEException)`
Gets the name of this queue.

6.554.1 Detailed Description

An interface encapsulating a provider-specific queue name. Messages sent to a **Queue** (p. 2674) are sent to a Single Subscriber on that **Queue** (p. 2674) **Destination** (p. 1480). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 2163) in a **Queue** (p. 2674) is not defined by the CMS API, consult your Provider documentation for this information.

Since:

1.0

6.554.2 Constructor & Destructor Documentation

6.554.2.1 virtual `cms::Queue::~Queue ()` [inline, virtual]

6.554.3 Member Function Documentation

6.554.3.1 virtual `std::string cms::Queue::getQueueName () const throw (CMSEException)` [pure virtual]

Gets the name of this queue.

Returns:

The queue name.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in `activemq::commands::ActiveMQQueue` (p. 422).

The documentation for this class was generated from the following file:

- `src/main/cms/Queue.h`

6.555 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p.2674) without removing them.

#include <src/main/cms/QueueBrowser.h> Inheritance diagram for cms::QueueBrowser:

Public Member Functions

- virtual **~QueueBrowser** ()
- virtual const **Queue** * **getQueue** () const =0 throw (cms::CMSEException)
- virtual std::string **getMessageSelector** () const =0 throw (cms::CMSEException)
- virtual std::vector< const **cms::Message** * > **getEnumeration** () const =0 throw (cms::CMSEException)

Gets an enumeration for browsing the current queue messages in the order they would be received.

6.555.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p.2674) without removing them. The **getEnumeration** method of this class returns a static snapshot of the **Queue** (p.2674) at the time the method is called. Since new Message's can be arriving and old Message's could expire the client should periodically refresh its view by calling **getEnumeration** again.

Since:

1.1

6.555.2 Constructor & Destructor Documentation

6.555.2.1 virtual cms::QueueBrowser::~~QueueBrowser () [inline, virtual]

6.555.3 Member Function Documentation

6.555.3.1 virtual std::vector<const cms::Message*>
cms::QueueBrowser::getEnumeration () const throw (
cms::CMSEException) [pure virtual]

Gets an enumeration for browsing the current queue messages in the order they would be received. The enumeration returned is a static view of the **Queue** (p.2674) and is not updated as new Messages arrive, the client should refresh its enumeration by calling this method again.

Returns:

an STL vector for browsing the messages.

Exceptions:

CMSEException (p. 1031) if an internal error occurs.

6.555.3.2 `virtual std::string cms::QueueBrowser::getMessageSelector () const
throw (cms::CMSEException) [pure virtual]`

Returns:

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions:

CMSEException (p. 1031) if an internal error occurs.

6.555.3.3 `virtual const Queue* cms::QueueBrowser::getQueue () const throw (
cms::CMSEException) [pure virtual]`

Returns:

the **Queue** (p. 2674) that this browser is listening on.

Exceptions:

CMSEException (p. 1031) if an internal error occurs.

The documentation for this class was generated from the following file:

- `src/main/cms/QueueBrowser.h`

6.556 decaf::util::Random Class Reference

Random (p. 2677) Value Generator which is used to generate a stream of pseudorandom numbers.

```
#include <src/main/decaf/util/Random.h>
```

Public Member Functions

- **Random** ()
Construct a random generator with the current time of day in milliseconds as the initial state.
- **Random** (unsigned long long seed)
*Construct a random generator with the given **seed** as the initial state.*
- bool **nextBoolean** ()
Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.
- void **nextBytes** (std::vector< unsigned char > &buf)
Modifies the byte array by a random sequence of bytes generated by this random number generator.
- double **nextDouble** ()
Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.
- float **nextFloat** ()
Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.
- double **nextGaussian** ()
*Pseudo-randomly generates (approximately) a normally distributed **double** value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.*
- int **nextInt** ()
*Generates a uniformly distributed 32-bit **int** value from the this random number sequence.*
- int **nextInt** (int n) throw (lang::exceptions::IllegalArgumentException)
*Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of **n** (exclusively).*
- long long **nextLong** ()
*Generates a uniformly distributed 64-bit **int** value from the this random number sequence.*
- void **setSeed** (unsigned long long seed)
*Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2, Section 3.2.1.**

Protected Member Functions

- virtual int **next** (int bits)
*Answers a pseudo-random uniformly distributed **int** value of the number of bits specified by the argument **bits** as described by Donald E.*

6.556.1 Detailed Description

Random (p. 2677) Value Generator which is used to generate a stream of pseudorandom numbers. The algorithms implemented by class **Random** (p. 2677) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since:

1.0

6.556.2 Constructor & Destructor Documentation

6.556.2.1 decaf::util::Random::Random ()

Construct a random generator with the current time of day in milliseconds as the initial state.

See also:

`setSeed` (p. 2681)

6.556.2.2 decaf::util::Random::Random (unsigned long long *seed*)

Construct a random generator with the given *seed* as the initial state.

Parameters:

seed the seed that will determine the initial state of this random number generator

See also:

`setSeed` (p. 2681)

6.556.3 Member Function Documentation

6.556.3.1 virtual int decaf::util::Random::next (int *bits*) [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument *bits* as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns:

`int` a pseudo-random generated `int` number

Parameters:

bits number of bits of the returned value

See also:

`nextBytes` (p. 2679)

`nextDouble` (p. 2679)

`nextFloat` (p. 2679)

`nextInt()` (p. 2680)

nextInt(int) (p. 2680)
nextGaussian (p. 2680)
nextLong (p. 2680)

6.556.3.2 bool decaf::util::Random::nextBoolean ()

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

Returns:

boolean a pseudo-random, uniformly distributed boolean value

6.556.3.3 void decaf::util::Random::nextBytes (std::vector< unsigned char > & buf)

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters:

buf non-null array to contain the new random bytes

See also:

next (p. 2678)

6.556.3.4 double decaf::util::Random::nextDouble ()

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

Returns:

double

See also:

nextFloat (p. 2679)

6.556.3.5 float decaf::util::Random::nextFloat ()

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

Returns:

float a random float number between 0.0 and 1.0

See also:

nextDouble (p. 2679)

6.556.3.6 double decaf::util::Random::nextGaussian ()

Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G. E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

Returns:

`double`

See also:

`nextDouble` (p. 2679)

6.556.3.7 int decaf::util::Random::nextInt (int n) throw (lang::exceptions::IllegalArgumentException)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

Returns:

`int`

Parameters:

`n` `int`

Exceptions:

IllegalArgumentException

6.556.3.8 int decaf::util::Random::nextInt ()

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

Returns:

`int` uniformly distributed `int` value

See also:

`next` (p. 2678)

`nextLong` (p. 2680)

6.556.3.9 long long decaf::util::Random::nextLong ()

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

Returns:

64-bit `int` random number

See also:

`next` (p. 2678)
`nextInt()` (p. 2680)
`nextInt(int)` (p. 2680)

6.556.3.10 void decaf::util::Random::setSeed (unsigned long long *seed*)

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters:

seed the seed that alters the state of the random number generator

See also:

`next` (p. 2678)
`Random()` (p. 2678)
`Random(long)`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Random.h`

6.557 activemq::transport::inactivity::ReadChecker Class Reference

Runnable class that is used by the {}.

#include <src/main/activemq/transport/inactivity/ReadChecker.h>Inheritance diagram for activemq::transport::inactivity::ReadChecker:

Public Member Functions

- **ReadChecker** (**InactivityMonitor** *parent)
- virtual ~**ReadChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.557.1 Detailed Description

Runnable class that is used by the {}.

See also:

InactivityMonitor (p. 1733)} class the check for timeouts related to **transport** (p. 79) reads.

Since:

3.1

6.557.2 Constructor & Destructor Documentation

6.557.2.1 **activemq::transport::inactivity::ReadChecker::ReadChecker** (**InactivityMonitor** * *parent*)

6.557.2.2 **virtual activemq::transport::inactivity::ReadChecker::~~ReadChecker** ()
[virtual]

6.557.3 Member Function Documentation

6.557.3.1 **virtual void activemq::transport::inactivity::ReadChecker::run** ()
[virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2816).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**ReadChecker.h**

6.558 decaf::io::Reader Class Reference

```
#include <src/main/decaf/io/Reader.h>
```

Public Member Functions

- virtual **~Reader** ()
- virtual void **setInputStream** (InputStream *is)=0
Sets the target input stream.
- virtual **InputStream * getInputStream** ()=0
Gets the target input stream.
- virtual std::size_t **read** (unsigned char *buffer, std::size_t count)=0 throw (IOException, lang::exceptions::NullPointerException)
Attempts to read an array of bytes from the stream.
- virtual unsigned char **readByte** ()=0 throw (IOException)
Attempts to read a byte from the input stream.

6.558.1 Constructor & Destructor Documentation

6.558.1.1 virtual decaf::io::Reader::~~Reader () [inline, virtual]

6.558.2 Member Function Documentation

6.558.2.1 virtual InputStream* decaf::io::Reader::getInputStream () [pure virtual]

Gets the target input stream.

6.558.2.2 virtual std::size_t decaf::io::Reader::read (unsigned char * *buffer*, std::size_t *count*) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]

Attempts to read an array of bytes from the stream.

Parameters:

- buffer* The target byte buffer.
count The number of bytes to read.

Returns:

The number of bytes read.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

6.558.2.3 `virtual unsigned char decaf::io::Reader::readByte () throw (IOException)` [pure virtual]

Attempts to read a byte from the input stream.

Returns:

The byte.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

6.558.2.4 `virtual void decaf::io::Reader::setInputStream (InputStream * is)` [pure virtual]

Sets the target input stream.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Reader.h`

6.559 decaf::nio::ReadOnlyBufferException Class Reference

#include <src/main/decaf/nio/ReadOnlyBufferException.h> Inheritance diagram for decaf::nio::ReadOnlyBufferException:

Public Member Functions

- **ReadOnlyBufferException** () throw ()
Default Constructor.
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const ReadOnlyBufferException &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ReadOnlyBufferException** (const std::exception *cause) throw ()
Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **ReadOnlyBufferException** * clone () const
Clones this exception.
- virtual ~**ReadOnlyBufferException** () throw ()

6.559.1 Constructor & Destructor Documentation

6.559.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException () throw () [inline]

Default Constructor.

6.559.1.2 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.559.1.3 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const ReadOnlyBufferException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.559.1.4 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.559.1.5 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.559.1.6 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.559.1.7 virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException
() throw () [inline, virtual]

6.559.2 Member Function Documentation

6.559.2.1 virtual ReadOnlyBufferException* de-
caf::nio::ReadOnlyBufferException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::UnsupportedOperationException** (p. 3305).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ReadOnlyBufferException.h**

6.560 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 2688) maintains a pair of associated locks, one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

Public Member Functions

- virtual **~ReadWriteLock** ()
- virtual **Lock & readLock** ()=0
Returns the lock used for reading.
- virtual **Lock & writeLock** ()=0
Returns the lock used for writing.

6.560.1 Detailed Description

A **ReadWriteLock** (p. 2688) maintains a pair of associated locks, one for read-only operations and one for writing. The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 2688) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 2017) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

* Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible. * Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency. * Determining whether the locks are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant? * Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

Since:

1.0

6.560.2 Constructor & Destructor Documentation

6.560.2.1 `virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock ()`
[inline, virtual]

6.560.3 Member Function Documentation

6.560.3.1 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock ()`
[pure virtual]

Returns the lock used for reading.

Returns:

the lock used for reading.

6.560.3.2 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock ()`
[pure virtual]

Returns the lock used for writing.

Returns:

the lock used for writing.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReadWriteLock.h`

6.561 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** *parent, **cms::Destination** *destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (**cms::Session** *session) throw (cms::CMSEException)
Execute any number of operations against the supplied CMS session.
- virtual **cms::Destination** * **getDestination** (**cms::Session** *session AMQCPP_UNUSED) throw (cms::CMSEException)
- **cms::Message** * **getMessage** ()

Protected Attributes

- **cms::Destination** * destination
- std::string selector
- bool noLocal
- **cms::Message** * message
- **CmsTemplate** * parent

6.561.1 Constructor & Destructor Documentation

6.561.1.1 **activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor** (**CmsTemplate** * parent, **cms::Destination** * destination, const std::string & selector, bool noLocal) [inline]

6.561.1.2 **virtual**
activemq::cmsutil::CmsTemplate::ReceiveExecutor::~~ReceiveExecutor ()
[inline, virtual]

6.561.2 Member Function Documentation

6.561.2.1 **virtual void** **activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms** (**cms::Session** * session) throw (cms::CMSEException) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters:

session the CMS Session

Exceptions:

cms::CMSEException (p. 1031) if thrown by CMS API methods

Implements `activemq::cmsutil::SessionCallback` (p. 2852).

6.561.2.2 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination (cms::Session *session AMQCPP_UNUSED) throw (cms::CMSException)` [inline, virtual]

6.561.2.3 `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage ()` [inline]

6.561.3 Field Documentation

6.561.3.1 `cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination` [protected]

6.561.3.2 `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message` [protected]

6.561.3.3 `bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal` [protected]

6.561.3.4 `CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent` [protected]

6.561.3.5 `std::string activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.562 `decaf::util::concurrent::locks::ReentrantLock` Class Reference

A reentrant mutual exclusion **Lock** (p. 2017) with extended capabilities.

#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h> Inheritance diagram for `decaf::util::concurrent::locks::ReentrantLock`:

Public Member Functions

- **ReentrantLock** ()
- virtual **~ReentrantLock** ()
- virtual void **lock** () throw (`decaf::lang::exceptions::RuntimeException`)
Acquires the lock.
- virtual void **lockInterruptibly** () throw (`decaf::lang::exceptions::RuntimeException`, `decaf::lang::exceptions::InterruptedException`)
Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** () throw (`decaf::lang::exceptions::RuntimeException`)
Acquires the lock only if it is not held by another thread at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit) throw (`decaf::lang::exceptions::RuntimeException`, `decaf::lang::exceptions::InterruptedException`)
Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** () throw (`decaf::lang::exceptions::RuntimeException`, `decaf::lang::exceptions::IllegalMonitorStateException`)
Attempts to release this lock.
- virtual **Condition** * **newCondition** () throw (`decaf::lang::exceptions::RuntimeException`, `decaf::lang::exceptions::UnsupportedOperationException`)
*Returns a **Condition** (p. 1118) instance for use with this **Lock** (p. 2017) instance.*
- int **getHoldCount** () const
Queries the number of holds on this lock by the current thread.
- bool **isHeldByCurrentThread** () const
Queries if this lock is held by the current thread.
- bool **isLocked** () const
Queries if this lock is held by any thread.
- bool **isFair** () const
Returns true if this lock has fairness set true.
- std::string **toString** () const
Returns a string identifying this lock, as well as its lock state.

6.562.1 Detailed Description

A reentrant mutual exclusion **Lock** (p.2017) with extended capabilities. A **ReentrantLock** (p.2692) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods **isHeldByCurrentThread()** (p.2694), and **getHoldCount()** (p.2693).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, locks favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair locks accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain locks and guarantee lack of starvation. Note however, that fairness of locks does not guarantee fairness of thread scheduling. Thus, one of many threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed tryLock method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:  
ReentrantLock (p.2692) lock; // ...  
  
public:  
  
void m() { lock.lock(); // block until condition holds  
  
try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p.2017) interface, this class defines methods **isLocked** and **getLockQueueLength**, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

Since:

1.0

6.562.2 Constructor & Destructor Documentation

6.562.2.1 decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ()

6.562.2.2 virtual decaf::util::concurrent::locks::ReentrantLock::~ReentrantLock ()
[virtual]

6.562.3 Member Function Documentation

6.562.3.1 int decaf::util::concurrent::locks::ReentrantLock::getHoldCount () const

Queries the number of holds on this lock by the current thread. A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of code should not be entered with the lock already held then we can assert that fact:

```
class X { private:
ReentrantLock (p. 2692) lock; // ...
public:
void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body } catch(...)
{ lock.unlock(); } } }
```

Returns:

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread

6.562.3.2 bool decaf::util::concurrent::locks::ReentrantLock::isFair () const

Returns true if this lock has fairness set true.

Returns:

true if this lock has fairness set true

6.562.3.3 bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread () const

Queries if this lock is held by the current thread. This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```
class X { private: ReentrantLock (p. 2692) lock = new ReentrantLock() (p. 2693); // ...
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }
```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```
class X { private: ReentrantLock (p. 2692) lock = new ReentrantLock() (p. 2693); // ...
public: void m() { assert !lock.isHeldByCurrentThread() (p. 2694); lock.lock(); try { // ...
method body } finally { lock.unlock(); } } }
```

Returns:

true if current thread holds this lock and false otherwise

6.562.3.4 bool decaf::util::concurrent::locks::ReentrantLock::isLocked () const

Queries if this lock is held by any thread. This method is designed for use in monitoring of the system state, not for synchronization control.

Returns:

true if any thread holds this lock and false otherwise

6.562.3.5 virtual void decaf::util::concurrent::locks::ReentrantLock::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Acquires the lock. Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 2018).

6.562.3.6 virtual void decaf::util::concurrent::locks::ReentrantLock::lockInterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException) [virtual]

Acquires the lock unless the current thread is interrupted. Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock,

then *InterruptedException* is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implements **decaf::util::concurrent::locks::Lock** (p. 2019).

6.562.3.7 `virtual Condition* decaf::util::concurrent::locks::ReentrantLock::newCondition()` `throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException)` [virtual]

Returns a **Condition** (p. 1118) instance for use with this **Lock** (p. 2017) instance. The returned **Condition** (p. 1118) instance supports the same usages as do the **Mutex** (p. 2385) Class' methods (wait, notify, and notifyAll).

* If this lock is not held when any of the **Condition** (p. 1118) waiting or signalling methods are called, then an `IllegalMonitorStateException` is thrown. * When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. * If a thread is interrupted while waiting then the wait will terminate, an `InterruptedException` will be thrown, and the thread's interrupted status will be cleared. * Waiting threads are signaled in FIFO order. * The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair locks favors those threads that have been waiting the longest.

Exceptions:

RuntimeException if an error occurs while creating the **Condition** (p. 1118).

UnsupportedOperationException if this **Lock** (p. 2017) implementation does not support conditions

Implements `decaf::util::concurrent::locks::Lock` (p. 2019).

6.562.3.8 `std::string decaf::util::concurrent::locks::ReentrantLock::toString()` `const`

Returns a string identifying this lock, as well as its lock state. The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

Returns:

a string identifying this lock, as well as its lock state

6.562.3.9 `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock(long long time, const TimeUnit & unit)` `throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)` [virtual]

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted. Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting for the lock. This is in contrast to the **tryLock()** (p. 2697) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread;
or * The specified waiting time elapses

If the lock is acquired then the value `true` is returned and the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value `false` is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

Parameters:

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns:

`true` if the lock was acquired and `false` if the waiting time elapsed before the lock was acquired

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements `decaf::util::concurrent::locks::Lock` (p. 2020).

6.562.3.10 virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Acquires the lock only if it is not held by another thread at the time of invocation. Acquires the lock if it is not held by another thread and returns immediately with the value `true`, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to `tryLock()` (p. 2697) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use `tryLock(0, TimeUnit.SECONDS)` (p. 3214) which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns `true`.

If the lock is held by another thread then this method will return immediately with the value `false`.

Returns:

`true` if the lock was acquired and `false` otherwise

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

Implements `decaf::util::concurrent::locks::Lock` (p. 2021).

6.562.3.11 `virtual void decaf::util::concurrent::locks::ReentrantLock::unlock
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Attempts to release this lock. If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then `IllegalMonitorStateException` is thrown.

Exceptions:

RuntimeException if an error occurs while acquiring the lock.

Implements `decaf::util::concurrent::locks::Lock` (p. 2021).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReentrantLock.h`

6.563 decaf::util::concurrent::RejectedExecutionException Class Reference

#include <src/main/decaf/util/concurrent/RejectedExecutionException.h> Inheritance diagram for decaf::util::concurrent::RejectedExecutionException:

Public Member Functions

- **RejectedExecutionException** () throw ()
Default Constructor.
- **RejectedExecutionException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()
Copy Constructor.
- **RejectedExecutionException** (const std::exception *cause) throw ()
Constructor.
- **RejectedExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RejectedExecutionException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **RejectedExecutionException** * clone () const
Clones this exception.
- virtual ~**RejectedExecutionException** () throw ()

6.563.1 Constructor & Destructor Documentation

6.563.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException () throw () [inline]

Default Constructor.

6.563.1.2 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.563.1.3 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const RejectedExecutionException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

6.563.1.4 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.563.1.5 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.563.1.6 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.563.1.7 **virtual**
 decaf::util::concurrent::RejectedExecutionException::~~RejectedExecutionException
 () throw () [inline, virtual]

6.563.2 Member Function Documentation

6.563.2.1 **virtual RejectedExecutionException* de-**
 caf::util::concurrent::RejectedExecutionException::clone ()
 const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new instance this exception type with a copy the current state.

Reimplemented from **decaf::lang::Exception** (p. 1577).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionException.h`

6.564 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

```
#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>
```

Public Member Functions

- virtual **~RejectedExecutionHandler** ()

6.564.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

Since:

1.0

6.564.2 Constructor & Destructor Documentation

- 6.564.2.1 **virtual**
decaf::util::concurrent::RejectedExecutionHandler::~RejectedExecutionHandler
() [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionHandler.h`

6.565 activemq::commands::RemoveInfo Class Reference

#include <src/main/activemq/commands/RemoveInfo.h> Inheritance diagram for activemq::commands::RemoveInfo:

Public Member Functions

- **RemoveInfo** ()
- virtual **~RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **RemoveInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getObjectId** () const
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &objectId)
- virtual long long **getLastDeliveredSequenceId** () const
- virtual void **setLastDeliveredSequenceId** (long long lastDeliveredSequenceId)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVEINFO** = 12

Protected Member Functions

- **RemoveInfo** (const **RemoveInfo** &)
- **RemoveInfo** & **operator=** (const **RemoveInfo** &)

Protected Attributes

- `Pointer< DataStructure > objectId`
- `long long lastDeliveredSequenceId`

6.565.1 Constructor & Destructor Documentation

6.565.1.1 `activemq::commands::RemoveInfo::RemoveInfo (const RemoveInfo &)`
[inline, protected]

6.565.1.2 `activemq::commands::RemoveInfo::RemoveInfo ()`

6.565.1.3 `virtual activemq::commands::RemoveInfo::~~RemoveInfo ()` [virtual]

6.565.2 Member Function Documentation

6.565.2.1 `virtual RemoveInfo* activemq::commands::RemoveInfo::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.565.2.2 `virtual void activemq::commands::RemoveInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.565.2.3 `virtual bool activemq::commands::RemoveInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.565.2.4 virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSetructure** (p. 1461) type copy.

Implements **activemq::commands::DataSetructure** (p. 1464).

6.565.2.5 virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceId () const [virtual]

6.565.2.6 virtual Pointer<DataSetructure>& activemq::commands::RemoveInfo::getObjectId () [virtual]

6.565.2.7 virtual const Pointer<DataSetructure>& activemq::commands::RemoveInfo::getObjectId () const [virtual]

6.565.2.8 virtual bool activemq::commands::RemoveInfo::isRemoveInfo () const [inline, virtual]

Returns:

an answer of true to the **isRemoveInfo()** (p. 2705) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 654).

6.565.2.9 RemoveInfo& activemq::commands::RemoveInfo::operator= (const RemoveInfo &) [inline, protected]

6.565.2.10 virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceId (long long *lastDeliveredSequenceId*) [virtual]

6.565.2.11 virtual void activemq::commands::RemoveInfo::setObjectId (const Pointer< DataSetructure > & *objectId*) [virtual]

6.565.2.12 virtual std::string activemq::commands::RemoveInfo::toString () const [virtual]

Returns a string containing the information for this **DataSetructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.565.2.13 `virtual Pointer<Command> activemq::commands::RemoveInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.565.3 Field Documentation

6.565.3.1 `const unsigned char activemq::commands::RemoveInfo::ID_ -
REMOVEINFO = 12` [static]

6.565.3.2 `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId`
[protected]

6.565.3.3 `Pointer<DataStructure> activemq::commands::RemoveInfo::objectId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveInfo.h`

6.566 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2707).

#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.566.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2707). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.566.2 Constructor & Destructor Documentation

6.566.2.1 `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

6.566.2.2 `virtual activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

6.566.3 Member Function Documentation

6.566.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.566.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.566.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.566.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.566.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.566.3.6 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.566.3.7 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**

6.567 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2711).

#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.567.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2711). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.567.2 Constructor & Destructor Documentation

6.567.2.1 `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

6.567.2.2 `virtual activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

6.567.3 Member Function Documentation

6.567.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.567.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.567.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.567.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.567.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.567.3.6 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.567.3.7 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**

6.568 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2715).

#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.568.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2715). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.568.2 Constructor & Destructor Documentation

6.568.2.1 `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

6.568.2.2 `virtual activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

6.568.3 Member Function Documentation

6.568.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.568.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.568.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.568.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.568.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.568.3.6 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.568.3.7 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**

6.569 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2719).

#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.569.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2719). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.569.2 Constructor & Destructor Documentation

6.569.2.1 `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

6.569.2.2 `virtual activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

6.569.3 Member Function Documentation

6.569.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.569.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.569.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.569.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.569.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.569.3.6 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.569.3.7 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveInfoMarshaller.h**

6.570 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2723).

#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.570.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2723). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.570.2 Constructor & Destructor Documentation

6.570.2.1 `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::RemoveInfoMarshaller()` [inline]

6.570.2.2 `virtual activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::~~RemoveInfoMarshaller()` [inline, virtual]

6.570.3 Member Function Documentation

6.570.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.570.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.570.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.570.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.570.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.570.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 682).

6.570.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 683).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h`

6.571 activemq::commands::RemoveSubscriptionInfo Class Reference

#include <src/main/activemq/commands/RemoveSubscriptionInfo.h> Inheritance diagram for activemq::commands::RemoveSubscriptionInfo:

Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **RemoveSubscriptionInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVESUBSCRIPTIONINFO** = 9

Protected Member Functions

- **RemoveSubscriptionInfo** (const **RemoveSubscriptionInfo** &)
- **RemoveSubscriptionInfo** & **operator=** (const **RemoveSubscriptionInfo** &)

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **subscriptionName**
- std::string **clientId**

6.571.1 Constructor & Destructor Documentation

- 6.571.1.1** **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** (const **RemoveSubscriptionInfo** &) [inline, protected]
- 6.571.1.2** **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** ()
- 6.571.1.3** **virtual**
activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo () [virtual]

6.571.2 Member Function Documentation

- 6.571.2.1** **virtual RemoveSubscriptionInfo*** **activemq::commands::RemoveSubscriptionInfo::cloneDataStructure** ()
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

- 6.571.2.2** **virtual void** **activemq::commands::RemoveSubscriptionInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.571.2.3 virtual bool activemq::commands::RemoveSubscriptionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

6.571.2.4 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () [virtual]**6.571.2.5 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () const [virtual]****6.571.2.6 virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () [virtual]****6.571.2.7 virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () const [virtual]****6.571.2.8 virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType () const [virtual]**

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.571.2.9** `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName ()`
[virtual]
- 6.571.2.10** `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName ()`
`const` [virtual]
- 6.571.2.11** `virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo () const` [inline, virtual]

Returns:

an answer of true to the `isRemoveSubscriptionInfo()` (p. 2730) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 654).

- 6.571.2.12** `RemoveSubscriptionInfo& activemq::commands::RemoveSubscriptionInfo::operator=`
`(const RemoveSubscriptionInfo &)` [inline, protected]
- 6.571.2.13** `virtual void activemq::commands::RemoveSubscriptionInfo::setClientId`
`(const std::string & clientId)` [virtual]
- 6.571.2.14** `virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId (const`
`Pointer< ConnectionId > & connectionId)` [virtual]
- 6.571.2.15** `virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName`
`(const std::string & subscriptionName)` [virtual]
- 6.571.2.16** `virtual std::string activemq::commands::RemoveSubscriptionInfo::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

- 6.571.2.17** `virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit`
`(activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.571.3 Field Documentation

6.571.3.1 `std::string activemq::commands::RemoveSubscriptionInfo::clientId`
[protected]

6.571.3.2 `Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId`
[protected]

6.571.3.3 `const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVE_SUBSCRIPTION_INFO = 9` [static]

6.571.3.4 `std::string activemq::commands::RemoveSubscriptionInfo::subscriptionName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveSubscriptionInfo.h`

6.572 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2732).

#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h> In diagram for activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual ~**RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.572.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2732).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.572.2 Constructor & Destructor Documentation

6.572.2.1 `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` `[inline]`

6.572.2.2 `virtual activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller()` `[inline, virtual]`

6.572.3 Member Function Documentation

6.572.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.572.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.572.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.572.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.572.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.572

activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

Class Reference

2739

```
6.572.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMars
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
(p. 682).

```
6.572.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnm
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
(p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**RemoveSubscriptionInfoMarshaller.h**

6.573 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2736).

#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h> In diagram for activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.573.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2736).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.573

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

2741

6.573.2 Constructor & Destructor Documentation

6.573.2.1 `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` [inline]

6.573.2.2 `virtual activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller()` [inline, virtual]

6.573.3 Member Function Documentation

6.573.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObject(const commands::DataStructure& dataStructure) const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.573.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.573.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::marshal(const commands::DataStructure& dataStructure, decaf::io::DataOutputStream& dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.573.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.573.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.573

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

2743

6.573.3.6 virtual void ac-

```
timemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMars  
(OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut,  
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
(p. 689).

6.573.3.7 virtual void ac-

```
timemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnm  
(OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
(p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveSubscriptionInfoMarshaller.h**

6.574 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2740).

#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h> In diagram for activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.574.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2740).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.574.2 Constructor & Destructor Documentation

6.574.2.1 `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` [inline]

6.574.2.2 `virtual activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller()` [inline, virtual]

6.574.3 Member Function Documentation

6.574.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.574.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.574.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.574.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.574.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.574

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

2747

```
6.574.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMars
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

```
6.574.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnm
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**

6.575 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2744).

#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h> In diagram for activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.575.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2744).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.575.2 Constructor & Destructor Documentation

6.575.2.1 `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller()` [inline]

6.575.2.2 `virtual activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller()` [inline, virtual]

6.575.3 Member Function Documentation

6.575.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.575.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.575.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.575.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.575.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.575

activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

Class Reference

2751

```
6.575.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMars
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
(p. 661).

```
6.575.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnm
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
(p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveSubscriptionInfoMarshaller.h**

6.576 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2748).

#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h> In diagram for activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.576.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2748).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.576.2 Constructor & Destructor Documentation

6.576.2.1 `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfo()` [inline]

6.576.2.2 `virtual activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfo()` [inline, virtual]

6.576.3 Member Function Documentation

6.576.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.576.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.576.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::marshal(commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.576.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.576.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.576

activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

Class Reference

2755

```
6.576.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMars
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
(p. 675).

```
6.576.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightUnm
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
(p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveSubscriptionInfoMarshaller.h**

6.577 activemq::commands::ReplayCommand Class Reference

#include <src/main/activemq/commands/ReplayCommand.h> Inheritance diagram for activemq::commands::ReplayCommand:

Public Member Functions

- **ReplayCommand** ()
- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ReplayCommand** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REPLAYCOMMAND** = 65

Protected Member Functions

- **ReplayCommand** (const **ReplayCommand** &)
- **ReplayCommand** & **operator=** (const **ReplayCommand** &)

Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

6.577.1 Constructor & Destructor Documentation

6.577.1.1 `activemq::commands::ReplayCommand::ReplayCommand (const ReplayCommand &) [inline, protected]`

6.577.1.2 `activemq::commands::ReplayCommand::ReplayCommand ()`

6.577.1.3 `virtual activemq::commands::ReplayCommand::~~ReplayCommand () [virtual]`

6.577.2 Member Function Documentation

6.577.2.1 `virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.577.2.2 `virtual void activemq::commands::ReplayCommand::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.577.2.3 `virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.577.2.4 `virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.577.2.5 `virtual int activemq::commands::ReplayCommand::getFirstNakNumber () const [virtual]`

6.577.2.6 `virtual int activemq::commands::ReplayCommand::getLastNakNumber () const [virtual]`

6.577.2.7 `ReplayCommand& activemq::commands::ReplayCommand::operator= (const ReplayCommand &) [inline, protected]`

6.577.2.8 `virtual void activemq::commands::ReplayCommand::setFirstNakNumber (int firstNakNumber) [virtual]`

6.577.2.9 `virtual void activemq::commands::ReplayCommand::setLastNakNumber (int lastNakNumber) [virtual]`

6.577.2.10 `virtual std::string activemq::commands::ReplayCommand::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.577.2.11 `virtual Pointer<Command> activemq::commands::ReplayCommand::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.577.3 Field Documentation

6.577.3.1 `int activemq::commands::ReplayCommand::firstNakNumber` [protected]

6.577.3.2 `const unsigned char activemq::commands::ReplayCommand::ID_-REPLAYCOMMAND = 65` [static]

6.577.3.3 `int activemq::commands::ReplayCommand::lastNakNumber` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ReplayCommand.h`

6.578 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2756).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.578.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2756). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.578.2 Constructor & Destructor Documentation

6.578.2.1 `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

6.578.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

6.578.3 Member Function Documentation

6.578.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.578.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.578.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.578.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.578.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.578.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.578.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h

6.579 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2760).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.579.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2760). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.579.2 Constructor & Destructor Documentation

6.579.2.1 `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

6.579.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

6.579.3 Member Function Documentation

6.579.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.579.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.579.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

6.579.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

6.579.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

6.579.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.579.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h

6.580 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2764).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.580.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2764). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.580.2 Constructor & Destructor Documentation

6.580.2.1 `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

6.580.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

6.580.3 Member Function Documentation

6.580.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.580.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.580.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.580.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.580.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.580.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.580.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h

6.581 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2768).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.581.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2768). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.581.2 Constructor & Destructor Documentation

6.581.2.1 `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::ReplayCommandMarshaller()` [inline]

6.581.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` [inline, virtual]

6.581.3 Member Function Documentation

6.581.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.581.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.581.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.581.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.581.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.581.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.581.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h

6.582 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2772).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.582.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2772). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.582.2 Constructor & Destructor Documentation

6.582.2.1 `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayCommandMarshaller()` `[inline]`

6.582.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~~ReplayCommandMarshaller()` `[inline, virtual]`

6.582.3 Member Function Documentation

6.582.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.582.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.582.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.582.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.582.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.582.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.582.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h

6.583 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveProducerExecutor:

Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** **action*, **CmsTemplate** **parent*, const std::string &*destinationName*)
- virtual ~**ResolveProducerExecutor** ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** **session*) throw (cms::CMSException)

6.583.1 Constructor & Destructor Documentation

6.583.1.1 **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor** (**ProducerCallback** * *action*, **CmsTemplate** * *parent*, const std::string & *destinationName*) [inline]

6.583.1.2 virtual **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~~ResolveProducerExecutor** () [inline, virtual]

6.583.2 Member Function Documentation

6.583.2.1 virtual **cms::Destination*** **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination** (**cms::Session** * *session*) throw (**cms::CMSException**) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.584 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor:

Public Member Functions

- **ResolveReceiveExecutor** (**CmsTemplate** ***parent**, const std::string &**selector**, bool **noLocal**, const std::string &**destinationName**)
- virtual ~**ResolveReceiveExecutor** ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** ***session**) throw (cms::CMSEException)

6.584.1 Constructor & Destructor Documentation

6.584.1.1 **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor** (**CmsTemplate** * *parent*, const std::string & *selector*, bool *noLocal*, const std::string & *destinationName*) [inline]

6.584.1.2 virtual **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~~ResolveReceiveExecutor** () [inline, virtual]

6.584.2 Member Function Documentation

6.584.2.1 virtual **cms::Destination*** **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination** (**cms::Session** * *session*) throw (cms::CMSEException) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.585 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>
```

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
*Destructor - calls **destroy**.*
- void **addConnection** (cms::Connection *connection)
Adds a connection so that its life will be managed by this object.
- void **addSession** (cms::Session *session)
Adds a session so that its life will be managed by this object.
- void **addDestination** (cms::Destination *dest)
Adds a destination so that its life will be managed by this object.
- void **addMessageProducer** (cms::MessageProducer *producer)
Adds a message producer so that its life will be managed by this object.
- void **addMessageConsumer** (cms::MessageConsumer *consumer)
Adds a message consumer so that its life will be managed by this object.
- void **destroy** () throw (cms::CMSException)
Closes and destroys the contained CMS resources.
- void **releaseAll** ()
Releases all of the contained resources so that this object will no longer control their lifetimes.

6.585.1 Detailed Description

Manages the lifecycle of a set of CMS resources. A call to **destroy** will close and destroy all of the contained resources in the appropriate manner.

6.585.2 Constructor & Destructor Documentation

6.585.2.1 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager** ()

6.585.2.2 **virtual**
activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager () [virtual]

Destructor - calls **destroy**.

6.585.3 Member Function Documentation

6.585.3.1 void activemq::cmsutil::ResourceLifecycleManager::addConnection
(cms::Connection * *connection*) [inline]

Adds a connection so that its life will be managed by this object.

Parameters:

connection the object to be managed

6.585.3.2 void activemq::cmsutil::ResourceLifecycleManager::addDestination
(cms::Destination * *dest*) [inline]

Adds a destination so that its life will be managed by this object.

Parameters:

dest the object to be managed

6.585.3.3 void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer
(cms::MessageConsumer * *consumer*) [inline]

Adds a message consumer so that its life will be managed by this object.

Parameters:

consumer the object to be managed

6.585.3.4 void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer
(cms::MessageProducer * *producer*) [inline]

Adds a message producer so that its life will be managed by this object.

Parameters:

producer the object to be managed

6.585.3.5 void activemq::cmsutil::ResourceLifecycleManager::addSession
(cms::Session * *session*) [inline]

Adds a session so that its life will be managed by this object.

Parameters:

session the object to be managed

6.585.3.6 void activemq::cmsutil::ResourceLifecycleManager::destroy () throw (cms::CMSException)

Closes and destroys the contained CMS resources.

Exceptions:

cms::CMSException (p. 1031) thrown if an error occurs.

6.585.3.7 void activemq::cmsutil::ResourceLifecycleManager::releaseAll ()

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ResourceLifecycleManager.h**

6.586 activemq::commands::Response Class Reference

#include <src/main/activemq/commands/Response.h> Inheritance diagram for activemq::commands::Response:

Public Member Functions

- **Response** ()
- virtual **~Response** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **Response * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual int **getCorrelationId** () const
- virtual void **setCorrelationId** (int correlationId)
- virtual bool **isResponse** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_RESPONSE** = 30

Protected Member Functions

- **Response** (const **Response** &)
- **Response & operator=** (const **Response** &)

Protected Attributes

- int **correlationId**

6.586.1 Constructor & Destructor Documentation

6.586.1.1 `activemq::commands::Response::Response (const Response &) [inline, protected]`

6.586.1.2 `activemq::commands::Response::Response ()`

6.586.1.3 `virtual activemq::commands::Response::~~Response () [virtual]`

6.586.2 Member Function Documentation

6.586.2.1 `virtual Response* activemq::commands::Response::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1357), `activemq::commands::DataResponse` (p. 1396), `activemq::commands::ExceptionResponse` (p. 1583), and `activemq::commands::IntegerResponse` (p. 1778).

6.586.2.2 `virtual void activemq::commands::Response::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

Reimplemented in `activemq::commands::DataArrayResponse` (p. 1357), `activemq::commands::DataResponse` (p. 1396), `activemq::commands::ExceptionResponse` (p. 1583), and `activemq::commands::IntegerResponse` (p. 1778).

6.586.2.3 `virtual bool activemq::commands::Response::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 651).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1357), **activemq::commands::DataResponse** (p. 1396), **activemq::commands::ExceptionResponse** (p. 1583), and **activemq::commands::IntegerResponse** (p. 1778).

6.586.2.4 `virtual int activemq::commands::Response::getCorrelationId () const [virtual]`

6.586.2.5 `virtual unsigned char activemq::commands::Response::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1358), **activemq::commands::DataResponse** (p. 1397), **activemq::commands::ExceptionResponse** (p. 1583), and **activemq::commands::IntegerResponse** (p. 1778).

6.586.2.6 `virtual bool activemq::commands::Response::isResponse () const [inline, virtual]`

Returns:

an answer of true to the **isResponse()** (p. 2783) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 654).

6.586.2.7 `Response& activemq::commands::Response::operator= (const Response &) [inline, protected]`

6.586.2.8 `virtual void activemq::commands::Response::setCorrelationId (int correlationId) [virtual]`

6.586.2.9 `virtual std::string activemq::commands::Response::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

Reimplemented in `activemq::commands::DataArrayResponse`
 (p. 1358), `activemq::commands::DataResponse` (p. 1397), `ac-`
`tivemq::commands::ExceptionResponse` (p. 1584), and `ac-`
`tivemq::commands::IntegerResponse` (p. 1779).

6.586.2.10 `virtual Pointer<Command> activemq::commands::Response::visit`
`(activemq::state::CommandVisitor * visitor) throw (`
`exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1067).

6.586.3 Field Documentation

6.586.3.1 `int activemq::commands::Response::correlationId [protected]`

6.586.3.2 `const unsigned char activemq::commands::Response::ID_RESPONSE =`
`30 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Response.h`

6.587 activemq::transport::mock::ResponseBuilder Class Reference

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

#include <src/main/activemq/transport/mock/ResponseBuilder.h> Inheritance diagram for activemq::transport::mock::ResponseBuilder:

Public Member Functions

- virtual **~ResponseBuilder** ()
- virtual **Pointer< Response > buildResponse** (const **Pointer< Command > &command**)=0
Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.
- virtual void **buildIncomingCommands** (const **Pointer< Command > &command**, **decaf::util::StlQueue< Pointer< Command > > &queue**)=0
*When called the **ResponseBuilder** (p. 2785) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

6.587.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

6.587.2 Constructor & Destructor Documentation

6.587.2.1 virtual **activemq::transport::mock::ResponseBuilder::~~ResponseBuilder** () [inline, virtual]

6.587.3 Member Function Documentation

6.587.3.1 virtual void **activemq::transport::mock::ResponseBuilder::buildIncomingCommands** (const **Pointer< Command > &command**, **decaf::util::StlQueue< Pointer< Command > > &queue**) [pure virtual]

When called the **ResponseBuilder** (p. 2785) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters:

- command* - The Command being sent to the Broker.
- queue* - Queue of Command sent back from the broker.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2466).

6.587.3.2 `virtual Pointer<Response> activemq::transport::mock::ResponseBuilder::buildResponse
(const Pointer< Command > & command) [pure virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters:

command - The command to build a response for

Returns:

A Response object pointer, or NULL if no response.

Implemented in `activemq::wireformat::openwire::OpenWireResponseBuilder` (p. 2467).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/ResponseBuilder.h`

6.588 activemq::transport::correlator::ResponseCorrelator Class Reference

This type of transport filter is responsible for correlating asynchronous responses with requests.

#include <src/main/activemq/transport/correlator/ResponseCorrelator.h> Inheritance diagram for activemq::transport::correlator::ResponseCorrelator:

Public Member Functions

- **ResponseCorrelator** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual ~**ResponseCorrelator** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given request to the server and waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given request to the server and waits for the response.
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
This is called in the context of the nested transport's reading thread.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **onTransportException** (**Transport** *source, const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.

6.588.1 Detailed Description

This type of transport filter is responsible for correlating asynchronous responses with requests. Non-response messages are simply sent directly to the CommandListener. It owns the transport that it

6.588.2 Constructor & Destructor Documentation

6.588.2.1 `activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator (const Pointer< Transport > & next)`

Constructor.

Parameters:

next the next transport in the chain

6.588.2.2 `virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator () [virtual]`

6.588.3 Member Function Documentation

6.588.3.1 `virtual void activemq::transport::correlator::ResponseCorrelator::close () throw (decaf::io::IOException) [virtual]`

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

IOException if errors occur.

Reimplemented from `activemq::transport::TransportFilter` (p. 3283).

6.588.3.2 `virtual void activemq::transport::correlator::ResponseCorrelator::onCommand (const Pointer< Command > & command) [virtual]`

This is called in the context of the nested transport's reading thread. In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters:

command the received from the nested transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3285).

6.588.3.3 `virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3285).

6.588.3.4 `virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException (Transport * source, const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters:

source The source of the exception

ex The exception.

6.588.3.5 `virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

Parameters:

command The request to send.

timeout The time to wait for a response.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3286).

6.588.3.6 `virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

Parameters:

command The request to send.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3287).

**6.588.3.7 virtual void activemq::transport::correlator::ResponseCorrelator::start ()
throw (decaf::io::IOException) [virtual]**

Starts this transport object and creates the thread for polling on the input stream for commands. If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

IOException if an error occurs or if this transport has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3287).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

6.589 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2791).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.589.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2791). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.589.2 Constructor & Destructor Documentation

6.589.2.1 `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller()` [inline]

6.589.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

6.589.3 Member Function Documentation

6.589.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1586), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1781).

6.589.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1586), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1781).

6.589.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1586), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1781).

6.589.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1377), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1587), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1782).

6.589.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1377), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1587), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1782).

6.589.3.6 virtual void `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 689).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1378), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1417), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1588), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1783).

```
6.589.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1378), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1417), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1588), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1783).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h`

6.590 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2796).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.590.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2796). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.590.2 Constructor & Destructor Documentation

- 6.590.2.1** `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller()` [inline]
- 6.590.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

6.590.3 Member Function Documentation

- 6.590.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1360), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1594), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1789).

- 6.590.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1360), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1594), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1789).

- 6.590.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 658).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1360), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1399), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1594), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1789).

6.590.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 659).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1361), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1400), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1595), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1790).

6.590.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 660).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1361), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1400), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1595), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1790).

6.590.3.6 virtual void `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 661).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1362), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1401), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1596), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1791).

```
6.590.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn,
 utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 662).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1362), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1401), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1596), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1791).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h`

6.591 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p.2801).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.591.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p.2801). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.591.2 Constructor & Destructor Documentation

6.591.2.1 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::ResponseMarshaller()` [inline]

6.591.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

6.591.3 Member Function Documentation

6.591.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1602), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1797).

6.591.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1602), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1797).

6.591.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1372), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1407), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1602), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1797).

6.591.3.4 virtual void `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1373), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1603), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1798).

6.591.3.5 virtual int `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1373), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1603), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1798).

6.591.3.6 virtual void `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal2` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 682).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1374), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1409), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1604), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1799).


```
6.591.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1374), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1409), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1604), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 1799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ResponseMarshaller.h**

6.592 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2806).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.592.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2806). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.592.2 Constructor & Destructor Documentation

6.592.2.1 `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller()` [inline]

6.592.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

6.592.3 Member Function Documentation

6.592.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1368), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1590), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1785).

6.592.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1368), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1590), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1785).

6.592.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1368), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1590), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1785).

6.592.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1369), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1404), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1591), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1786).

6.592.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1369), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1404), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1591), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1786).

6.592.3.6 virtual void `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 668).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1370), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1405), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1592), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1787).

```
6.592.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn,
 utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 669).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1370), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1405), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1592), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1787).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h`

6.593 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p.2811).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.593.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p.2811). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.593.2 Constructor & Destructor Documentation

6.593.2.1 `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::ResponseMarshaller()` [inline]

6.593.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::~~ResponseMarshaller()` [inline, virtual]

6.593.3 Member Function Documentation

6.593.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1364), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1598), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1793).

6.593.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1364), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1598), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1793).

6.593.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1364), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1598), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1793).

6.593.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1365), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1412), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1599), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1794).

6.593.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1365), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1412), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1599), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1794).

6.593.3.6 virtual void `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal2` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 675).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1366), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1413), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1600), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1795).

6.593.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1366), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1413), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1600), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 1795).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ResponseMarshaller.h**

6.594 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

#include <src/main/decaf/lang/Runnable.h>Inheritance diagram for decaf::lang::Runnable:

Public Member Functions

- virtual `~Runnable()`
- virtual void `run()`=0

Run method - called by the Thread class in the context of the thread.

6.594.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

6.594.2 Constructor & Destructor Documentation

6.594.2.1 virtual `decaf::lang::Runnable::~~Runnable()` [inline, virtual]

6.594.3 Member Function Documentation

6.594.3.1 virtual void `decaf::lang::Runnable::run()` [pure virtual]

Run method - called by the Thread class in the context of the thread.

Implemented in `activemq::threads::CompositeTaskRunner` (p. 1093), `activemq::threads::DedicatedTaskRunner` (p. 1473), `activemq::transport::inactivity::ReadChecker` (p. 2682), `activemq::transport::inactivity::WriteChecker` (p. 3403), and `activemq::transport::IOTransport` (p. 1828).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Runnable.h`

6.595 decaf::lang::Runtime Class Reference

#include <src/main/decaf/lang/Runtime.h> Inheritance diagram for decaf::lang::Runtime:

Public Member Functions

- virtual `~Runtime ()`

Static Public Member Functions

- static `Runtime * getRuntime ()`
*Gets the single instance of the Decaf **Runtime** (p. 2817) for this Process.*
- static void `initializeRuntime (int argc, char **argv)`
Initialize the Decaf Library passing it the args that were passed to the application at startup.
- static void `initializeRuntime ()`
Initialize the Decaf Library.
- static void `shutdownRuntime ()`
Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

6.595.1 Constructor & Destructor Documentation

6.595.1.1 virtual `decaf::lang::Runtime::~~Runtime ()` [inline, virtual]

6.595.2 Member Function Documentation

6.595.2.1 static `Runtime* decaf::lang::Runtime::getRuntime ()` [static]

Gets the single instance of the Decaf **Runtime** (p. 2817) for this Process.

Returns:

pointer to the single Decaf **Runtime** (p. 2817) instance that exists for this process

6.595.2.2 static void `decaf::lang::Runtime::initializeRuntime ()` [static]

Initialize the Decaf Library.

Exceptions:

runtime_error if the library is already initialized or an error occurs during initialization.

6.596 decaf::lang::exceptions::RuntimeException Class Reference

#include <src/main/decaf/lang/exceptions/RuntimeException.h> Inheritance diagram for decaf::lang::exceptions::RuntimeException:

Public Member Functions

- **RuntimeException** () throw ()
Default Constructor.
- **RuntimeException** (const **Exception** &ex) throw ()
Conversion Constructor from some other ActiveMQException.
- **RuntimeException** (const **RuntimeException** &ex) throw ()
Copy Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RuntimeException** (const std::exception *cause) throw ()
Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **RuntimeException** * clone () const
Clones this exception.
- virtual ~**RuntimeException** () throw ()

6.596.1 Constructor & Destructor Documentation

6.596.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException () throw () [inline]

Default Constructor.

6.596.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const Exception & ex) throw () [inline]

Conversion Constructor from some other ActiveMQException.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.596.1.3 decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1574) whose data is to be copied into this one.

6.596.1.4 decaf::lang::exceptions::RuntimeException::RuntimeException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.596.1.5 decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.596.1.6 decaf::lang::exceptions::RuntimeException::RuntimeException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.596.1.7 virtual decaf::lang::exceptions::RuntimeException::~~RuntimeException
() throw () [inline, virtual]

6.596.2 Member Function Documentation

6.596.2.1 virtual RuntimeException* decaf::lang::exceptions::RuntimeException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p.1574) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1577).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**RuntimeException.h**

6.597 decaf::security_provider::SecurityProvider Class Reference

```
#include <src/main/decaf/security_provider/SecurityProvider.h>
```

Public Member Functions

- virtual `~SecurityProvider()`
- virtual `X500Principal * createX500Principal (const std::string &name)=0`
- virtual `X500Principal * createX500Principal (InputStream &is)=0`
- virtual `X500Principal * createX500Principal (unsigned char *buffer, int offset, int len)=0`

6.597.1 Constructor & Destructor Documentation

6.597.1.1 virtual `decaf::security_provider::SecurityProvider::~~SecurityProvider()`
[inline, virtual]

6.597.2 Member Function Documentation

6.597.2.1 virtual `X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (unsigned char * buffer, int offset, int len)` [pure virtual]

6.597.2.2 virtual `X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (InputStream & is)` [pure virtual]

6.597.2.3 virtual `X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (const std::string & name)` [pure virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security_provider/SecurityProvider.h`

6.598 decaf::security_provider::SecurityProviderMap Class Reference

Lookup Map for Connector Factories.

```
#include <src/main/decaf/security_provider/SecurityProviderMap.h>
```

Public Member Functions

- void **registerSecurityProvider** (const std::string &name, **SecurityProvider** *provider)
Registers a new provider with this map.
- void **unregisterSecurityProvider** (const std::string &name)
Unregisters a provider from this map.
- **SecurityProvider** * **lookup** (const std::string &name)
Lookup the named provider in the Map.
- std::size_t **getSecurityProviderNames** (std::vector< std::string > &providers)
Fetch a list of provider names that this Map contains.

Static Public Member Functions

- static **SecurityProviderMap** * **getInstance** ()
Gets a singleton instance of this class.

6.598.1 Detailed Description

Lookup Map for Connector Factories. Use the Connector name to find the associated factory. This class does not take ownership of the stored factories, they must be deallocated somewhere.

6.598.2 Member Function Documentation

6.598.2.1 static **SecurityProviderMap*** decaf::security_provider::SecurityProviderMap::getInstance () [static]

Gets a singleton instance of this class.

Referenced by decaf::security_provider::SecurityProviderRegistrar::SecurityProviderRegistrar(), and decaf::security_provider::SecurityProviderRegistrar::~~SecurityProviderRegistrar().

6.598.2.2 std::size_t decaf::security_provider::SecurityProviderMap::getSecurityProviderNames (std::vector< std::string > & providers)

Fetch a list of provider names that this Map contains.

Parameters:

providers A vector object to receive the list

Returns:

count of providers.

6.598.2.3 `SecurityProvider* decaf::security_ -
provider::SecurityProviderMap::lookup (const std::string
& name)`

Lookup the named provider in the Map.

Parameters:

name the provider name to lookup

Returns:

the provider associated with the name, or NULL

6.598.2.4 `void decaf::security_ -
provider::SecurityProviderMap::registerSecurityProvider
(const std::string & name, SecurityProvider * provider)`

Registers a new provider with this map.

Parameters:

name A name to associate the provider with
provider the provider object to store in the map.

6.598.2.5 `void decaf::security_ -
provider::SecurityProviderMap::unregisterSecurityProvider (const
std::string & name)`

Unregisters a provider from this map.

Parameters:

name the name of the provider to remove

The documentation for this class was generated from the following file:

- `src/main/decaf/security_provider/SecurityProviderMap.h`

6.599 decaf::security_provider::SecurityProviderRegistrar Class Reference

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

```
#include <src/main/decaf/security_provider/SecurityProviderRegistrar.h>
```

Public Member Functions

- **SecurityProviderRegistrar** (const std::string &name, **SecurityProvider** *provider, bool manageLifetime=true)
Creates a registrar and registers the provider with the provider map.
- virtual ~**SecurityProviderRegistrar** ()
Unregisters the provider from the provider map and destroys it if `manageLifetime` is set.
- virtual **SecurityProvider** * **getProvider** ()
get a reference to the factory that this class is holding

6.599.1 Detailed Description

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

6.599.2 Constructor & Destructor Documentation

6.599.2.1 decaf::security_provider::SecurityProviderRegistrar::SecurityProviderRegistrar (const std::string & *name*, **SecurityProvider** * *provider*, bool *manageLifetime* = true) [inline]

Creates a registrar and registers the provider with the provider map.

Parameters:

name name of the provider to register

provider the provider object

manageLifetime boolean indicating if this object manages the lifetime of the factory that is being registered.

References decaf::security_provider::SecurityProviderMap::getInstance().

6.599.2.2 virtual decaf::security_provider::SecurityProviderRegistrar::~SecurityProviderRegistrar () [inline, virtual]

Unregisters the provider from the provider map and destroys it if `manageLifetime` is set.

References decaf::security_provider::SecurityProviderMap::getInstance().

6.599.3 Member Function Documentation

6.599.3.1 `virtual SecurityProvider* decaf::security_
provider::SecurityProviderRegistrar::getProvider ()`
 `[inline, virtual]`

get a reference to the factory that this class is holding

Returns:

reference to a factory class

The documentation for this class was generated from the following file:

- `src/main/decaf/security_provider/SecurityProviderRegistrar.h`

6.600 decaf::util::concurrent::Semaphore Class Reference

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

Public Member Functions

- **Semaphore** (int permits)
*Creates a **Semaphore** (p. 2827) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)
*Creates a **Semaphore** (p. 2827) with the given number of permits and the given fairness setting.*
- virtual **~Semaphore** ()
- void **acquire** () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.
- void **acquireUninterruptibly** () throw (decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, blocking until one is available.
- bool **tryAcquire** () throw (decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, only if one is available at the time of invocation.
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.
- void **release** () throw (decaf::lang::exceptions::RuntimeException)
Releases a permit, returning it to the semaphore.
- void **acquire** (int permits) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.
- void **acquireUninterruptibly** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, blocking until all are available.
- bool **tryAcquire** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.
- bool **tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

- void **release** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Releases the given number of permits, returning them to the semaphore.

- int **availablePermits** () const

Returns the current number of permits available in this semaphore.

- int **drainPermits** () throw (decaf::lang::exceptions::RuntimeException)

Acquires and returns all permits that are immediately available.

- bool **isFair** () const

- std::string **toString** () const

Returns a string identifying this semaphore, as well as its state.

6.600.1 Detailed Description

A counting semaphore. Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 2830) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 2833) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 2827) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
```

```
static const int MAX_AVAILABLE = 100; Semaphore (p. 2827) available;
```

```
std::vector<std::string> items; std::vector<bool> used;
```

```
Mutex (p. 2385) lock;
```

```
public:
```

```
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE ); items.resize(
MAX_AVAILABLE ); }
```

```
std::string getItem() throws InterruptedException { available.acquire(); return getNextAvail-
ableItem(); }
```

```
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }
```

```
std::string getNextAvailableItem() {
```

```
synchronized( &lock ) (p. 4126) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( !used[i] )
{ used[i] = true; return items[i]; } }
```

```
return std::string(); // not reached }
```

```
bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 4126) { for( int i =
0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) { used[i] = false; return
true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back

to the pool and a permit is returned to the semaphore, allowing another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 2830) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this way, the binary semaphore has the property (unlike many **Lock** (p. 2023) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 2830) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific internal points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

Since:

1.0

6.600.2 Constructor & Destructor Documentation

6.600.2.1 decaf::util::concurrent::Semaphore::Semaphore (int *permits*)

Creates a **Semaphore** (p. 2827) with the given number of permits and nonfair fairness setting.

Parameters:

permits the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

6.600.2.2 decaf::util::concurrent::Semaphore::Semaphore (int *permits*, bool *fair*)

Creates a **Semaphore** (p. 2827) with the given number of permits and the given fairness setting.

Parameters:

permits the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

fair true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

6.600.2.3 `virtual decaf::util::concurrent::Semaphore::~~Semaphore ()` [virtual]

6.600.3 Member Function Documentation

6.600.3.1 `void decaf::util::concurrent::Semaphore::acquire (int permits)`
`throw (decaf::lang::exceptions::InterruptedException,`
`decaf::lang::exceptions::IllegalArgumentException,`
`decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted. Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to `release()` (p. 2833).

Parameters:

permits the number of permits to acquire.

Exceptions:

InterruptedException if the current thread is interrupted.

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.2 `void decaf::util::concurrent::Semaphore::acquire ()`
`throw (decaf::lang::exceptions::InterruptedException,`
`decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted. Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes the `release()` (p. 2833) method for this semaphore and the current thread is next to be assigned a permit; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Exceptions:

InterruptedException - if the current thread is interrupted.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.3 void decaf::util::concurrent::Semaphore::acquireUninterruptibly (int *permits*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Acquires the given number of permits from this semaphore, blocking until all are available. Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

Parameters:

permits the number of permits to acquire.

Exceptions:

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.4 void decaf::util::concurrent::Semaphore::acquireUninterruptibly () throw (decaf::lang::exceptions::RuntimeException)

Acquires a permit from this semaphore, blocking until one is available. Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes the **release()** (p. 2833) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

Exceptions:

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.5 int decaf::util::concurrent::Semaphore::availablePermits () const

Returns the current number of permits available in this semaphore. This method is typically used for debugging and testing purposes.

Returns:

the number of permits available in this semaphore

6.600.3.6 int decaf::util::concurrent::Semaphore::drainPermits () throw (decaf::lang::exceptions::RuntimeException)

Acquires and returns all permits that are immediately available.

Returns:

the number of permits acquired

Exceptions:

RuntimeException if an unexpected error occurs while draining the **Semaphore** (p. 2827).

6.600.3.7 bool decaf::util::concurrent::Semaphore::isFair () const**Returns:**

true if this semaphore has fairness set true

**6.600.3.8 void decaf::util::concurrent::Semaphore::release (int *permits*)
 throw (decaf::lang::exceptions::IllegalArgumentException,
 decaf::lang::exceptions::RuntimeException)**

Releases the given number of permits, returning them to the semaphore. Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

Parameters:

permits the number of permits to release

Exceptions:

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while releasing the **Semaphore** (p. 2827).

6.600.3.9 void decaf::util::concurrent::Semaphore::release () throw (decaf::lang::exceptions::RuntimeException)

Releases a permit, returning it to the semaphore. Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 2830). Correct usage of a semaphore is established by programming convention in the application.

Exceptions:

RuntimeException if an unexpected error occurs while releasing the **Semaphore** (p. 2827).

6.600.3.10 std::string decaf::util::concurrent::Semaphore::toString () const

Returns a string identifying this semaphore, as well as its state. The state, in brackets, includes the String "Permits =" followed by the number of permits.

Returns:

a string identifying this semaphore, as well as its state

6.600.3.11 bool decaf::util::concurrent::Semaphore::tryAcquire (int *permits*, long long *timeout*, const TimeUnit & *unit*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted. Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 2833).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 2833).

Parameters:

permits the number of permits to acquire
timeout the maximum amount of time to wait to acquire the permits.
unit the units that the timeout param represents.

Returns:

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

Exceptions:

IllegalArgumentException if the permits argument is negative.
RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.12 `bool decaf::util::concurrent::Semaphore::tryAcquire (int permits)
 throw (decaf::lang::exceptions::IllegalArgumentException,
 decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation. Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to tryAcquire will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use tryAcquire(permits, 0, **TimeUnit.SECONDS** (p. 3214)) which is almost equivalent (it also detects interruption).

Parameters:

permits the number of permits to acquire

Returns:

true if the permits were acquired and false otherwise.

Exceptions:

IllegalArgumentException if the permits argument is negative.
RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.13 `bool decaf::util::concurrent::Semaphore::tryAcquire
 (long long timeout, const TimeUnit & unit) throw
 (decaf::lang::exceptions::InterruptedException,
 decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted. Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes the **release()** (p.2833) method for this semaphore and the current thread is next to be assigned a permit; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses.

If a permit is acquired then the value `true` is returned.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while waiting to acquire a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value `false` is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters:

timeout the maximum time to wait for a permit

unit the time unit of the timeout argument

Returns:

`true` if a permit was acquired and `false` if the waiting time elapsed before a permit was acquired

Exceptions:

InterruptedException if the current thread is interrupted.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

6.600.3.14 `bool decaf::util::concurrent::Semaphore::tryAcquire () throw (decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, only if one is available at the time of invocation. Acquires a permit, if one is available and returns immediately, with the value `true`, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value `false`.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p.2835) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use `tryAcquire(0, TimeUnit.SECONDS)` (p.3214)) which is almost equivalent (it also detects interruption).

Returns:

`true` if a permit was acquired and `false` otherwise

Exceptions:

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2827).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Semaphore.h`

6.601 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

#include <src/main/activemq/cmsutil/CmsTemplate.h> Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

Public Member Functions

- **SendExecutor** (**MessageCreator** *messageCreator, **CmsTemplate** *parent)
- virtual **~SendExecutor** ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer) throw (**cms::CMSException**)

Execute an action given a session and producer.

6.601.1 Constructor & Destructor Documentation

6.601.1.1 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** (**MessageCreator** * *messageCreator*, **CmsTemplate** * *parent*) [inline]

6.601.1.2 **virtual activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor** () [inline, virtual]

6.601.2 Member Function Documentation

6.601.2.1 **virtual void activemq::cmsutil::CmsTemplate::SendExecutor::doInCms** (**cms::Session** * *session*, **cms::MessageProducer** * *producer*) throw (**cms::CMSException**) [inline, virtual]

Execute an action given a session and producer.

Parameters:

session the CMS Session

producer the CMS Producer

Exceptions:

cms::CMSException (p. 1031) if thrown by CMS API methods

Implements **activemq::cmsutil::ProducerCallback** (p. 2603).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.602 decaf::net::ServerSocket Class Reference

A server socket class (for testing purposes).

```
#include <src/main/decaf/net/ServerSocket.h>
```

Public Types

- typedef apr_socket_t * **SocketHandle**
- typedef apr_sockaddr_t * **SocketAddress**

Public Member Functions

- **ServerSocket** ()
Constructor.
- virtual ~**ServerSocket** ()
Destructor.
- virtual void **bind** (const char *host, int port) throw (SocketException)
Bind and listen to given IP/dns and port.
- virtual void **bind** (const char *host, int port, int backlog) throw (SocketException)
Bind and listen to given IP/dns and port.
- virtual **Socket** * **accept** () throw (SocketException)
Blocks until a client connects to the bound socket.
- virtual void **close** () throw (lang::Exception)
Closes the server socket.
- virtual bool **isBound** () const

6.602.1 Detailed Description

A server socket class (for testing purposes).

6.602.2 Member Typedef Documentation

6.602.2.1 typedef apr_sockaddr_t* decaf::net::ServerSocket::SocketAddress

6.602.2.2 typedef apr_socket_t* decaf::net::ServerSocket::SocketHandle

6.602.3 Constructor & Destructor Documentation

6.602.3.1 decaf::net::ServerSocket::ServerSocket ()

Constructor. Creates a non-bound server socket.

6.602.3.2 virtual decaf::net::ServerSocket::~~ServerSocket () [virtual]

Destructor. Releases socket handle if `close()` (p. 2838) hasn't been called.

6.602.4 Member Function Documentation**6.602.4.1 virtual Socket* decaf::net::ServerSocket::accept () throw (SocketException) [virtual]**

Blocks until a client connects to the bound socket.

Returns:

new socket. Never returns NULL.

6.602.4.2 virtual void decaf::net::ServerSocket::bind (const char * *host*, int *port*, int *backlog*) throw (SocketException) [virtual]

Bind and listen to given IP/dns and port.

Parameters:

host IP address or host name.

port TCP port between 1..655535

backlog Size of listen backlog.

6.602.4.3 virtual void decaf::net::ServerSocket::bind (const char * *host*, int *port*) throw (SocketException) [virtual]

Bind and listen to given IP/dns and port.

Parameters:

host IP address or host name.

port TCP port between 1..655535

6.602.4.4 virtual void decaf::net::ServerSocket::close () throw (lang::Exception) [virtual]

Closes the server socket.

6.602.4.5 virtual bool decaf::net::ServerSocket::isBound () const [virtual]**Returns:**

true if the server socket is bound.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocket.h`

6.603 cms::Session Class Reference

A **Session** (p.2839) object is a single-threaded context for producing and consuming messages.

#include <src/main/cms/Session.h> Inheritance diagram for cms::Session:

Public Types

- enum **AcknowledgeMode** {
AUTO_ACKNOWLEDGE, **DUPS_OK_ACKNOWLEDGE**, **CLIENT_-ACKNOWLEDGE**, **SESSION_TRANSACTIONED**,
INDIVIDUAL_ACKNOWLEDGE }

Public Member Functions

- virtual **~Session** ()
- virtual void **close** ()=0 throw (CMSEException)
Closes this session as well as any active child consumers or producers.
- virtual void **commit** ()=0 throw (CMSEException)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()=0 throw (CMSEException)
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()=0 throw (CMSEException)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination)=0 throw (CMSEException)
*Creates a **MessageConsumer** (p. 2214) for the specified destination.*
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector)=0 throw (CMSEException)
*Creates a **MessageConsumer** (p. 2214) for the specified destination, using a message selector.*
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector, bool noLocal)=0 throw (CMSEException)
*Creates a **MessageConsumer** (p. 2214) for the specified destination, using a message selector.*
- virtual **MessageConsumer** * **createDurableConsumer** (const **Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw (CMSEException)
*Creates a durable subscriber to the specified topic, using a **Message** (p. 2163) selector.*
- virtual **MessageProducer** * **createProducer** (const **Destination** *destination)=0 throw (CMSEException)

*Creates a **MessageProducer** (p. 2332) to send messages to the specified destination.*

- virtual **QueueBrowser** * **createBrowser** (const **cms::Queue** *queue)=0 throw (CMSException)

*Creates a new **QueueBrowser** (p. 2675) to peek at Messages on the given **Queue** (p. 2674).*

- virtual **QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector)=0 throw (CMSException)

*Creates a new **QueueBrowser** (p. 2675) to peek at Messages on the given **Queue** (p. 2674).*

- virtual **Queue** * **createQueue** (const std::string &queueName)=0 throw (CMSException)

*Creates a queue identity given a **Queue** (p. 2674) name.*

- virtual **Topic** * **createTopic** (const std::string &topicName)=0 throw (CMSException)

*Creates a topic identity given a **Queue** (p. 2674) name.*

- virtual **TemporaryQueue** * **createTemporaryQueue** ()=0 throw (CMSException)

*Creates a **TemporaryQueue** (p. 3166) object.*

- virtual **TemporaryTopic** * **createTemporaryTopic** ()=0 throw (CMSException)

*Creates a **TemporaryTopic** (p. 3168) object.*

- virtual **Message** * **createMessage** ()=0 throw (CMSException)

*Creates a new **Message** (p. 2163).*

- virtual **BytesMessage** * **createBytesMessage** ()=0 throw (CMSException)

*Creates a **BytesMessage** (p. 938).*

- virtual **BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize)=0 throw (CMSException)

*Creates a **BytesMessage** (p. 938) and sets the payload to the passed value.*

- virtual **StreamMessage** * **createStreamMessage** ()=0 throw (CMSException)

*Creates a new **StreamMessage** (p. 3081).*

- virtual **TextMessage** * **createTextMessage** ()=0 throw (CMSException)

*Creates a new **TextMessage** (p. 3170).*

- virtual **TextMessage** * **createTextMessage** (const std::string &text)=0 throw (CMSException)

*Creates a new **TextMessage** (p. 3170) and set the text to the value given.*

- virtual **MapMessage** * **createMapMessage** ()=0 throw (CMSException)

*Creates a new **MapMessage** (p. 2106).*

- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw (CMSException)

Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () const =0 throw (CMSEException)
*Gets if the Session is a Transacted **Session** (p. 2839).*
- virtual void **unsubscribe** (const std::string &name)=0 throw (CMSEException)
Unsubscribes a durable subscription that has been created by a client.

6.603.1 Detailed Description

A **Session** (p.2839) object is a single-threaded context for producing and consuming messages. A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for TemporaryTopics and TemporaryQueues.
- It provides a way to create **Queue** (p. 2674) or **Topic** (p. 3215) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 2214) until a message arrives. The thread may then use one or more of the Session's MessageProducers.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's **close** method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 2839) method that can be called concurrently.
- Invoking any other **Session** (p. 2839) method on a closed session must throw an **IllegalStateException** (p. 1729). Closing a closed session must not throw any exceptions.

Transacted Sessions

When a **Session** (p. 2839) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 2839) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 2839). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 2332) this implies that all messages sent by the producer are not sent to the Provider until the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 2214) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 2163) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 2163).

Since:

1.0

6.603.2 Member Enumeration Documentation

6.603.2.1 enum cms::Session::AcknowledgeMode

Enumerator:

AUTO_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

DUPS_OK_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

CLIENT_ACKNOWLEDGE With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

SESSION_TRANSACTED Messages will be consumed when the transaction commits.

INDIVIDUAL_ACKNOWLEDGE **Message** (p. 2163) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

6.603.3 Constructor & Destructor Documentation

6.603.3.1 virtual cms::Session::~~Session () [inline, virtual]

6.603.4 Member Function Documentation

6.603.4.1 virtual void cms::Session::close () throw (CMSEException) [pure virtual]

Closes this session as well as any active child consumers or producers.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implements **cms::Closeable** (p. 1021).

Implemented in **activemq::cmsutil::PooledSession** (p. 2507), and **activemq::core::ActiveMQSession** (p. 447).

6.603.4.2 `virtual void cms::Session::commit () throw (CMSExcption) [pure virtual]`

Commits all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSExcption (p. 1031) - If an internal error occurs.

IllegalStateException (p. 1729) - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2507), and **activemq::core::ActiveMQSession** (p. 447).

Referenced by **activemq::cmsutil::PooledSession::commit()**.

6.603.4.3 `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * queue, const std::string & selector) throw (CMSExcption) [pure virtual]`

Creates a new **QueueBrowser** (p. 2675) to peek at Messages on the given **Queue** (p. 2674).

Parameters:

queue the **Queue** (p. 2674) to browse

selector the **Message** (p. 2163) selector to filter which messages are browsed.

Returns:

New **QueueBrowser** (p. 2675) that is owned by the caller.

Exceptions:

CMSExcption (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2508), and **activemq::core::ActiveMQSession** (p. 447).

6.603.4.4 `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * queue) throw (CMSExcption) [pure virtual]`

Creates a new **QueueBrowser** (p. 2675) to peek at Messages on the given **Queue** (p. 2674).

Parameters:

queue the **Queue** (p. 2674) to browse

Returns:

New **QueueBrowser** (p. 2675) that is owned by the caller.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2508), and **activemq::core::ActiveMQSession** (p. 448).

6.603.4.5 `virtual BytesMessage* cms::Session::createBytesMessage (const unsigned char * bytes, std::size_t bytesSize) throw (CMSEException) [pure virtual]`

Creates a **BytesMessage** (p. 938) and sets the payload to the passed value.

Parameters:

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2509), and **activemq::core::ActiveMQSession** (p. 448).

6.603.4.6 `virtual BytesMessage* cms::Session::createBytesMessage () throw (CMSEException) [pure virtual]`

Creates a **BytesMessage** (p. 938).

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2509), and **activemq::core::ActiveMQSession** (p. 449).

Referenced by `activemq::cmsutil::PooledSession::createBytesMessage()`.

6.603.4.7 `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination, const std::string & selector, bool noLocal) throw (CMSEException) [pure virtual]`

Creates a **MessageConsumer** (p. 2214) for the specified destination, using a message selector.

Parameters:

destination the **Destination** (p. 1480) that this consumer receiving messages for.

selector the **Message** (p. 2163) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new **MessageConsumer** (p. 2214) that is owned by the caller (caller deletes)

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if an invalid destination is specified.

InvalidSelectorException (p. 1816) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2510), and **activemq::core::ActiveMQSession** (p. 449).

6.603.4.8 virtual **MessageConsumer*** **cms::Session::createConsumer** (const **Destination** * *destination*, const std::string & *selector*) throw (**CMSEException**) [pure virtual]

Creates a **MessageConsumer** (p. 2214) for the specified destination, using a message selector.

Parameters:

destination the **Destination** (p. 1480) that this consumer receiving messages for.

selector the **Message** (p. 2163) Selector to use

Returns:

pointer to a new **MessageConsumer** (p. 2214) that is owned by the caller (caller deletes)

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if an invalid destination is specified.

InvalidSelectorException (p. 1816) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2510), and **activemq::core::ActiveMQSession** (p. 449).

6.603.4.9 virtual **MessageConsumer*** **cms::Session::createConsumer** (const **Destination** * *destination*) throw (**CMSEException**) [pure virtual]

Creates a **MessageConsumer** (p. 2214) for the specified destination.

Parameters:

destination the **Destination** (p. 1480) that this consumer receiving messages for.

Returns:

pointer to a new **MessageConsumer** (p. 2214) that is owned by the caller (caller deletes)

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2511), and **activemq::core::ActiveMQSession** (p. 450).

Referenced by **activemq::cmsutil::PooledSession::createConsumer()**.

6.603.4.10 **virtual MessageConsumer* cms::Session::createDurableConsumer (const Topic * *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = false) throw (CMSEException)** [pure virtual]

Creates a durable subscriber to the specified topic, using a **Message** (p. 2163) selector. Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters:

destination the topic to subscribe to

name The name used to identify the subscription

selector the **Message** (p. 2163) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new durable **MessageConsumer** (p. 2214) that is owned by the caller (caller deletes)

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if an invalid destination is specified.

InvalidSelectorException (p. 1816) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2511), and **activemq::core::ActiveMQSession** (p. 450).

Referenced by **activemq::cmsutil::PooledSession::createDurableConsumer()**.

6.603.4.11 **virtual MapMessage* cms::Session::createMapMessage () throw (CMSEException)** [pure virtual]

Creates a new **MapMessage** (p. 2106).

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2512), and **activemq::core::ActiveMQSession** (p. 450).

Referenced by **activemq::cmsutil::PooledSession::createMapMessage()**.

6.603.4.12 `virtual Message* cms::Session::createMessage () throw (CMSException) [pure virtual]`

Creates a new **Message** (p. 2163).

Exceptions:

CMSException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2512), and **activemq::core::ActiveMQSession** (p. 451).

Referenced by **activemq::cmsutil::PooledSession::createMessage()**.

6.603.4.13 `virtual MessageProducer* cms::Session::createProducer (const Destination * destination) throw (CMSException) [pure virtual]`

Creates a **MessageProducer** (p. 2332) to send messages to the specified destination.

Parameters:

destination the **Destination** (p. 1480) to send on

Returns:

New **MessageProducer** (p. 2332) that is owned by the caller.

Exceptions:

CMSException (p. 1031) - If an internal error occurs.

InvalidDestinationException (p. 1809) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2512), and **activemq::core::ActiveMQSession** (p. 451).

Referenced by **activemq::cmsutil::PooledSession::createProducer()**.

6.603.4.14 `virtual Queue* cms::Session::createQueue (const std::string & queueName) throw (CMSException) [pure virtual]`

Creates a queue identity given a **Queue** (p. 2674) name.

Parameters:

queueName the name of the new **Queue** (p. 2674)

Returns:

new **Queue** (p. 2674) pointer that is owned by the caller.

Exceptions:

CMSException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2513), and **activemq::core::ActiveMQSession** (p. 451).

Referenced by **activemq::cmsutil::PooledSession::createQueue()**.

6.603.4.15 `virtual StreamMessage* cms::Session::createStreamMessage () throw (CMSEException) [pure virtual]`

Creates a new **StreamMessage** (p. 3081).

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2513), and **activemq::core::ActiveMQSession** (p. 451).

Referenced by **activemq::cmsutil::PooledSession::createStreamMessage()**.

6.603.4.16 `virtual TemporaryQueue* cms::Session::createTemporaryQueue () throw (CMSEException) [pure virtual]`

Creates a **TemporaryQueue** (p. 3166) object.

Returns:

new **TemporaryQueue** (p. 3166) pointer that is owned by the caller.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2514), and **activemq::core::ActiveMQSession** (p. 452).

Referenced by **activemq::cmsutil::PooledSession::createTemporaryQueue()**.

6.603.4.17 `virtual TemporaryTopic* cms::Session::createTemporaryTopic () throw (CMSEException) [pure virtual]`

Creates a **TemporaryTopic** (p. 3168) object.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2514), and **activemq::core::ActiveMQSession** (p. 452).

Referenced by **activemq::cmsutil::PooledSession::createTemporaryTopic()**.

6.603.4.18 `virtual TextMessage* cms::Session::createTextMessage (const std::string & text) throw (CMSEException) [pure virtual]`

Creates a new **TextMessage** (p. 3170) and set the text to the value given.

Parameters:

text the initial text for the message

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2514), and **activemq::core::ActiveMQSession** (p. 452).

6.603.4.19 virtual TextMessage* cms::Session::createTextMessage () throw (CMSEException) [pure virtual]

Creates a new **TextMessage** (p. 3170).

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2514), and **activemq::core::ActiveMQSession** (p. 452).

Referenced by **activemq::cmsutil::PooledSession::createTextMessage()**.

6.603.4.20 virtual Topic* cms::Session::createTopic (const std::string & topicName) throw (CMSEException) [pure virtual]

Creates a topic identity given a **Queue** (p. 2674) name.

Parameters:

topicName the name of the new **Topic** (p. 3215)

Returns:

new **Topic** (p. 3215) pointer that is owned by the caller.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2515), and **activemq::core::ActiveMQSession** (p. 453).

Referenced by **activemq::cmsutil::PooledSession::createTopic()**.

6.603.4.21 virtual AcknowledgeMode cms::Session::getAcknowledgeMode () const throw (CMSEException) [pure virtual]

Returns the acknowledgment mode of the session.

Returns:

the Sessions Acknowledge Mode

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2515), and **activemq::core::ActiveMQSession** (p. 454).

Referenced by **activemq::cmsutil::PooledSession::getAcknowledgeMode()**.

6.603.4.22 **virtual bool cms::Session::isTransacted () const throw (CMSException)** [pure virtual]

Gets if the Sessions is a Transacted **Session** (p. 2839).

Returns:

transacted true - false.

Exceptions:

CMSException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2516), and **activemq::core::ActiveMQSession** (p. 456).

Referenced by **activemq::cmsutil::PooledSession::isTransacted()**.

6.603.4.23 **virtual void cms::Session::recover () throw (CMSException)** [pure virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException (p. 1729) - if the method is called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2516), and **activemq::core::ActiveMQSession** (p. 456).

Referenced by **activemq::cmsutil::PooledSession::recover()**.

6.603.4.24 **virtual void cms::Session::rollback () throw (CMSException)** [pure virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

IllegalStateException (p. 1729) - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2517), and **activemq::core::ActiveMQSession** (p. 457).

Referenced by **activemq::cmsutil::PooledSession::rollback()**.

6.603.4.25 virtual void cms::Session::unsubscribe (const std::string & name) throw (CMSEException) [pure virtual]

Unsubscribes a durable subscription that has been created by a client. This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 2214) or **Subscriber** for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters:

name The name used to identify this subscription

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2517), and **activemq::core::ActiveMQSession** (p. 458).

Referenced by **activemq::cmsutil::PooledSession::unsubscribe()**.

The documentation for this class was generated from the following file:

- **src/main/cms/Session.h**

6.604 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

#include <src/main/activemq/cmsutil/SessionCallback.h> Inheritance diagram for activemq::cmsutil::SessionCallback:

Public Member Functions

- virtual `~SessionCallback ()`
- virtual void `doInCms (cms::Session *session)=0` throw (cms::CMSEException)

Execute any number of operations against the supplied CMS session.

6.604.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.

6.604.2 Constructor & Destructor Documentation

- 6.604.2.1** virtual `activemq::cmsutil::SessionCallback::~~SessionCallback ()` [inline, virtual]

6.604.3 Member Function Documentation

- 6.604.3.1** virtual void `activemq::cmsutil::SessionCallback::doInCms (cms::Session * session)` throw (cms::CMSEException) [pure virtual]

Execute any number of operations against the supplied CMS session.

Parameters:

session the CMS Session

Exceptions:

cms::CMSEException (p. 1031) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::ProducerExecutor` (p. 2604), and `activemq::cmsutil::CmsTemplate::ReceiveExecutor` (p. 2690).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionCallback.h`

6.605 activemq::commands::SessionId Class Reference

#include <src/main/activemq/commands/SessionId.h> Inheritance diagram for activemq::commands::SessionId:

Public Types

- typedef decaf::lang::PointerComparator< SessionId > COMPARATOR

Public Member Functions

- SessionId ()
- SessionId (const SessionId &other)
- SessionId (const ConnectionId *connectionId, long long sessionId)
- SessionId (const ProducerId *producerId)
- SessionId (const ConsumerId *consumerId)
- virtual ~SessionId ()
- virtual unsigned char getDataStructureType () const
Get the unique identifier that this object and its own Marshaler share.
- virtual SessionId * cloneDataStructure () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void copyDataStructure (const DataStructure *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string toString () const
Returns a string containing the information for this DataStructure (p. 1461) such as its type and value of its elements.
- virtual bool equals (const DataStructure *value) const
Compares the DataStructure (p. 1461) passed in to this one, and returns if they are equivalent.
- const Pointer< ConnectionId > & getParentId () const
- virtual const std::string & getConnectionId () const
- virtual std::string & getConnectionId ()
- virtual void setConnectionId (const std::string &connectionId)
- virtual long long getValue () const
- virtual void setValue (long long value)
- virtual int compareTo (const SessionId &value) const
- virtual bool equals (const SessionId &value) const
- virtual bool operator== (const SessionId &value) const
- virtual bool operator< (const SessionId &value) const
- SessionId & operator= (const SessionId &other)

Static Public Attributes

- static const unsigned char **ID_SESSIONID** = 121

Protected Attributes

- std::string **connectionId**
- long long **value**

6.605.1 Member Typedef Documentation

- 6.605.1.1** `typedef decaf::lang::PointerComparator<SessionId>`
 `activemq::commands::SessionId::COMPARATOR`

6.605.2 Constructor & Destructor Documentation

- 6.605.2.1** `activemq::commands::SessionId::SessionId ()`
- 6.605.2.2** `activemq::commands::SessionId::SessionId (const SessionId & other)`
- 6.605.2.3** `activemq::commands::SessionId::SessionId (const ConnectionId *
 connectionId, long long sessionId)`
- 6.605.2.4** `activemq::commands::SessionId::SessionId (const ProducerId *
 producerId)`
- 6.605.2.5** `activemq::commands::SessionId::SessionId (const ConsumerId *
 consumerId)`
- 6.605.2.6** `virtual activemq::commands::SessionId::~~SessionId ()` [virtual]

6.605.3 Member Function Documentation

- 6.605.3.1** `virtual SessionId* activemq::commands::SessionId::cloneDataStructure ()`
 `const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

- 6.605.3.2** `virtual int activemq::commands::SessionId::compareTo (const SessionId
 & value) const` [virtual]
- 6.605.3.3** `virtual void activemq::commands::SessionId::copyDataStructure (const
 DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.605.3.4 `virtual bool activemq::commands::SessionId::equals (const SessionId & value) const` [virtual]

6.605.3.5 `virtual bool activemq::commands::SessionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.605.3.6 `virtual std::string& activemq::commands::SessionId::getConnectionId ()` [virtual]

6.605.3.7 `virtual const std::string& activemq::commands::SessionId::getConnectionId () const` [virtual]

Referenced by `activemq::commands::ConsumerId::ConsumerId()`, and `activemq::commands::ProducerId::ProducerId()`.

6.605.3.8 `virtual unsigned char activemq::commands::SessionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.605.3.9 `const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId () const`

6.605.3.10 `virtual long long activemq::commands::SessionId::getValue () const` [virtual]

Referenced by `activemq::commands::ConsumerId::ConsumerId()`, and `activemq::commands::ProducerId::ProducerId()`.

- 6.605.3.11 `virtual bool activemq::commands::SessionId::operator< (const SessionId & value) const [virtual]`
- 6.605.3.12 `SessionId& activemq::commands::SessionId::operator= (const SessionId & other)`
- 6.605.3.13 `virtual bool activemq::commands::SessionId::operator== (const SessionId & value) const [virtual]`
- 6.605.3.14 `virtual void activemq::commands::SessionId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.605.3.15 `virtual void activemq::commands::SessionId::setValue (long long value) [virtual]`
- 6.605.3.16 `virtual std::string activemq::commands::SessionId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.718).

6.605.4 Field Documentation

- 6.605.4.1 `std::string activemq::commands::SessionId::connectionId [protected]`
- 6.605.4.2 `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.605.4.3 `long long activemq::commands::SessionId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

6.606 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2857).

#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.606.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2857). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.606.2 Constructor & Destructor Documentation

6.606.2.1 `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::SessionIdMarshaller()` [inline]

6.606.2.2 `virtual activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

6.606.3 Member Function Documentation

6.606.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.606.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.606.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.606.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.606.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.606.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.606.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h

6.607 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2861).

#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.607.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2861). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.607.2 Constructor & Destructor Documentation

6.607.2.1 `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller()` [inline]

6.607.2.2 `virtual activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

6.607.3 Member Function Documentation

6.607.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.607.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.607.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.607.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.607.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.607.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.607.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionIdMarshaller.h**

6.608 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2865).

#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.608.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2865). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.608.2 Constructor & Destructor Documentation

6.608.2.1 `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller()` [inline]

6.608.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

6.608.3 Member Function Documentation

6.608.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.608.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.608.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.608.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.608.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.608.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.608.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h

6.609 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2869).

#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.609.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2869). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.609.2 Constructor & Destructor Documentation

6.609.2.1 `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller()` [inline]

6.609.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

6.609.3 Member Function Documentation

6.609.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.609.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.609.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.609.3.4 virtual void **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.609.3.5 virtual int **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.609.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.609.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h

6.610 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2873).

#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.610.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2873). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.610.2 Constructor & Destructor Documentation

6.610.2.1 `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::SessionIdMarshaller()` [inline]

6.610.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::~~SessionIdMarshaller()` [inline, virtual]

6.610.3 Member Function Documentation

6.610.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.610.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.610.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.610.3.4 virtual void **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.610.3.5 virtual int **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.610.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.610.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SessionIdMarshaller.h**

6.611 activemq::commands::SessionInfo Class Reference

#include <src/main/activemq/commands/SessionInfo.h> Inheritance diagram for activemq::commands::SessionInfo:

Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **SessionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- virtual const **Pointer**< **SessionId** > & **getSessionId** () const
- virtual **Pointer**< **SessionId** > & **getSessionId** ()
- virtual void **setSessionId** (const **Pointer**< **SessionId** > &sessionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SESSIONINFO** = 4

Protected Member Functions

- **SessionInfo** (const **SessionInfo** &)
- **SessionInfo** & **operator=** (const **SessionInfo** &)

Protected Attributes

- `Pointer< SessionId > sessionId`

6.611.1 Constructor & Destructor Documentation

6.611.1.1 `activemq::commands::SessionInfo::SessionInfo (const SessionInfo &)`
[inline, protected]

6.611.1.2 `activemq::commands::SessionInfo::SessionInfo ()`

6.611.1.3 `virtual activemq::commands::SessionInfo::~~SessionInfo ()` [virtual]

6.611.2 Member Function Documentation

6.611.2.1 `virtual SessionInfo* activemq::commands::SessionInfo::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.611.2.2 `virtual void activemq::commands::SessionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.611.2.3 `virtual bool activemq::commands::SessionInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.611.2.4 `unsigned int activemq::commands::SessionInfo::getAckMode () const`
[inline]

6.611.2.5 `virtual unsigned char activemq::commands::SessionInfo::getDataStructureType ()`
`const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

6.611.2.6 `virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ()`
[virtual]

6.611.2.7 `virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId () const`
[virtual]

6.611.2.8 `SessionInfo& activemq::commands::SessionInfo::operator= (const SessionInfo &)` [inline, protected]

6.611.2.9 `void activemq::commands::SessionInfo::setAckMode (unsigned int mode)`
[inline]

6.611.2.10 `virtual void activemq::commands::SessionInfo::setSessionId (const Pointer< SessionId > & sessionId)` [virtual]

6.611.2.11 `virtual std::string activemq::commands::SessionInfo::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.611.2.12 `virtual Pointer<Command> activemq::commands::SessionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1067).

6.611.3 Field Documentation

6.611.3.1 `const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4` [static]

6.611.3.2 `Pointer<SessionId> activemq::commands::SessionInfo::sessionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`

6.612 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2881).

#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.612.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2881). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.612.2 Constructor & Destructor Documentation

6.612.2.1 `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

6.612.2.2 `virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

6.612.3 Member Function Documentation

6.612.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.612.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.612.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 665).

6.612.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 666).

6.612.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 667).

6.612.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.612.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**

6.613 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2885).

#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.613.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2885). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.613.2 Constructor & Destructor Documentation

6.613.2.1 `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

6.613.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

6.613.3 Member Function Documentation

6.613.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.613.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.613.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.613.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.613.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.613.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.613.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h

6.614 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2889).

#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.614.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2889). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.614.2 Constructor & Destructor Documentation

6.614.2.1 `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

6.614.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

6.614.3 Member Function Documentation

6.614.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.614.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.614.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.614.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.614.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.614.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.614.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h

6.615 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2893).

#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.615.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2893). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.615.2 Constructor & Destructor Documentation

6.615.2.1 `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

6.615.2.2 `virtual activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

6.615.3 Member Function Documentation

6.615.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.615.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.615.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.615.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.615.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.615.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.615.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**SessionInfoMarshaller.h**

6.616 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2897).

#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.616.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2897). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.616.2 Constructor & Destructor Documentation

6.616.2.1 `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshaller()` [inline]

6.616.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~~SessionInfoMarshaller()` [inline, virtual]

6.616.3 Member Function Documentation

6.616.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.616.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.616.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.616.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.616.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.616.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.616.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h

6.617 activemq::cmsutil::SessionPool Class Reference

A pool of CMS sessions from the same connection and with the same acknowledge mode.

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

Public Member Functions

- **SessionPool** (**cms::Connection** *connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** *resourceLifecycleManager)
Constructs a session pool.
- virtual **~SessionPool** ()
Destroys the pooled session objects, but not the underlying session resources.
- virtual **PooledSession** * **takeSession** () throw (cms::CMSException)
Takes a session from the pool, creating one if necessary.
- virtual void **returnSession** (**PooledSession** *session)
Returns a session to the pool.
- **ResourceLifecycleManager** * **getResourceLifecycleManager** ()

6.617.1 Detailed Description

A pool of CMS sessions from the same connection and with the same acknowledge mode. Internal session resources are managed through a provided **ResourceLifecycleManager** (p. 2778), not by this pool. This class is thread-safe.

6.617.2 Constructor & Destructor Documentation

6.617.2.1 activemq::cmsutil::SessionPool::SessionPool (**cms::Connection** * connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** * resourceLifecycleManager)

Constructs a session pool.

Parameters:

connection the connection to be used for creating all sessions.

ackMode the acknowledge mode to be used for all sessions

resourceLifecycleManager the object responsible for managing the lifecycle of any allocated **cms::Session** (p. 2839) resources.

6.617.2.2 virtual activemq::cmsutil::SessionPool::~SessionPool () [virtual]

Destroys the pooled session objects, but not the underlying session resources. That is the job of the **ResourceLifecycleManager** (p. 2778).

6.617.3 Member Function Documentation

6.617.3.1 ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager()
() [inline]

6.617.3.2 virtual void activemq::cmsutil::SessionPool::returnSession (PooledSession* *session*) [virtual]

Returns a session to the pool.

Parameters:

session the session to be returned.

6.617.3.3 virtual PooledSession* activemq::cmsutil::SessionPool::takeSession ()
throw (cms::CMSEException) [virtual]

Takes a session from the pool, creating one if necessary.

Returns:

the pooled session object

Exceptions:

cms::CMSEException (p. 1031) if an error occurred

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/SessionPool.h

6.618 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

Public Member Functions

- **SessionState** (const **Pointer**< **SessionInfo** > &info)
- virtual ~**SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer**< **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

6.618.1 Constructor & Destructor Documentation

6.618.1.1 `activemq::state::SessionState::SessionState (const Pointer< SessionInfo > & info)`

6.618.1.2 `virtual activemq::state::SessionState::~~SessionState ()` [virtual]

6.618.2 Member Function Documentation

6.618.2.1 `void activemq::state::SessionState::addConsumer (const Pointer< ConsumerInfo > & info)` [inline]

6.618.2.2 `void activemq::state::SessionState::addProducer (const Pointer< ProducerInfo > & info)` [inline]

6.618.2.3 `void activemq::state::SessionState::checkShutdown ()` const

6.618.2.4 `Pointer<ConsumerState> activemq::state::SessionState::getConsumerState (const Pointer< ConsumerId > & id)` [inline]

6.618.2.5 `std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates ()` const [inline]

6.618.2.6 `const Pointer<SessionInfo> activemq::state::SessionState::getInfo ()` const [inline]

6.618.2.7 `Pointer<ProducerState> activemq::state::SessionState::getProducerState (const Pointer< ProducerId > & id)` [inline]

6.618.2.8 `std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates ()` const [inline]

6.618.2.9 `Pointer<ConsumerState> activemq::state::SessionState::removeConsumer (const Pointer< ConsumerId > & id)` [inline]

6.618.2.10 `Pointer<ProducerState> activemq::state::SessionState::removeProducer (const Pointer< ProducerId > & id)` [inline]

6.618.2.11 `void activemq::state::SessionState::shutdown ()` [inline]

6.618.2.12 `std::string activemq::state::SessionState::toString ()` const

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

6.619 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

#include <src/main/decaf/util/Set.h> Inheritance diagram for decaf::util::Set< E >:

Public Member Functions

- virtual `~Set ()`

6.619.1 Detailed Description

`template<typename E> class decaf::util::Set< E >`

A collection that contains no duplicate elements. More formally, sets contain no pair of elements `e1` and `e2` such that `e1 == e2`, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since:

1.0

6.619.2 Constructor & Destructor Documentation

6.619.2.1 `template<typename E> virtual decaf::util::Set< E >::~~Set () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Set.h`

6.620 decaf::lang::Short Class Reference

#include <src/main/decaf/lang/Short.h> Inheritance diagram for decaf::lang::Short:

Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const
*Compares this **Short** (p. 2906) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Short** &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const short &s) const
*Compares this **Short** (p. 2906) instance with another.*
- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const short &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Short** (p. 2906).*
- static short **reverseBytes** (short value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.
- static short **parseShort** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed short in the radix specified by the second argument.
- static short **parseShort** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal short.
- static **Short valueOf** (short value)
*Returns a **Short** (p. 2906) instance representing the specified short value.*
- static **Short valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Short** (p. 2906) object holding the value given by the specified std::string.*
- static **Short valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Short** (p. 2906) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 16
Size of this objects primitive type in bits.
- static const short **MAX_VALUE** = (short)0x7FFF
Max Value for this Object's primitive type.
- static const short **MIN_VALUE** = (short)0x8000
Max Value for this Object's primitive type.

6.620.1 Constructor & Destructor Documentation

6.620.1.1 decaf::lang::Short::Short (short value)

Parameters:

value - short to wrap

6.620.1.2 `decaf::lang::Short::Short (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters:

value - string value to convert to short and wrap

Exceptions:

NumberFormatException

6.620.1.3 `virtual decaf::lang::Short::~~Short () [inline, virtual]`

6.620.2 Member Function Documentation

6.620.2.1 `virtual unsigned char decaf::lang::Short::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2432).

6.620.2.2 `virtual int decaf::lang::Short::compareTo (const short & s) const [virtual]`

Compares this **Short** (p. 2906) instance with another.

Parameters:

s - the **Short** (p. 2906) instance to be compared

Returns:

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< short >` (p. 1083).

6.620.2.3 `virtual int decaf::lang::Short::compareTo (const Short & s) const [virtual]`

Compares this **Short** (p. 2906) instance with another.

Parameters:

s - the **Short** (p. 2906) instance to be compared

Returns:

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Short** > (p.1083).

6.620.2.4 static Short decaf::lang::Short::decode (const std::string & value) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Short** (p. 2906). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p.2912) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Short** (p.2906) object containing the decoded value

Exceptions:

NumberFormatException if the string is not formatted correctly.

6.620.2.5 virtual double decaf::lang::Short::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.620.2.6 bool decaf::lang::Short::equals (const short & s) const [inline, virtual]**Returns:**

true if the two **Short** (p. 2906) Objects have the same value.

Implements **decaf::lang::Comparable**< **short** > (p.1084).

6.620.2.7 bool decaf::lang::Short::equals (const Short & s) const [inline, virtual]**Returns:**

true if the two **Short** (p. 2906) Objects have the same value.

Implements **decaf::lang::Comparable**< **Short** > (p.1084).

6.620.2.8 `virtual float decaf::lang::Short::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.620.2.9 `virtual int decaf::lang::Short::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.620.2.10 `virtual long long decaf::lang::Short::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2433).

6.620.2.11 `virtual bool decaf::lang::Short::operator< (const short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< short >** (p. 1084).

6.620.2.12 `virtual bool decaf::lang::Short::operator< (const Short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Short** > (p.1084).

6.620.2.13 **virtual bool decaf::lang::Short::operator==(const short & s) const**
 [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **short** > (p.1085).

6.620.2.14 **virtual bool decaf::lang::Short::operator==(const Short & s) const**
 [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Short** > (p.1085).

6.620.2.15 **static short decaf::lang::Short::parseShort(const std::string & s) throw**
 (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal short. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseShort(const std::string, int)` method.

Parameters:

s - String to convert to a short

Returns:

the converted short value

Exceptions:

NumberFormatException if the string is not a short.

6.620.2.16 `static short decaf::lang::Short::parseShort (const std::string & s, int radix) throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed short in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 985) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:
* The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 989) or larger than **Character.MAX_RADIX** (p. 989). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type short.

Parameters:

s - the String containing the short representation to be parsed
radix - the radix to be used while parsing s

Returns:

the short represented by the string argument in the specified radix.

Exceptions:

NumberFormatException - If String does not contain a parsable short.

6.620.2.17 `static short decaf::lang::Short::reverseBytes (short value) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

Parameters:

value - the short whose bytes we are to reverse

Returns:

the reversed short.

6.620.2.18 `virtual short decaf::lang::Short::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2434).

6.620.2.19 `static std::string decaf::lang::Short::toString (short value) [static]`

Returns:

a string representing the primitive value as Base 10

6.620.2.20 `std::string decaf::lang::Short::toString () const`**Returns:**

this **Short** (p. 2906) Object as a String Representation

6.620.2.21 `static Short decaf::lang::Short::valueOf (const std::string & value, int radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Short** (p. 2906) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the `parseShort(std::string, int)` method. The result is a **Short** (p. 2906) object that represents the short value specified by the string.

Parameters:

value - `std::string` to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Short** (p. 2906) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid short.

6.620.2.22 `static Short decaf::lang::Short::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]`

Returns a **Short** (p. 2906) object holding the value given by the specified `std::string`. The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the `parseShort(std::string)` method. The result is a **Short** (p. 2906) object that represents the short value specified by the string.

Parameters:

value - `std::string` to parse as base 10

Returns:

new **Short** (p. 2906) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal short.

6.620.2.23 `static Short decaf::lang::Short::valueOf (short value) [static]`

Returns a **Short** (p. 2906) instance representing the specified short value.

Parameters:

value - the short to wrap

Returns:

the new **Short** (p. 2906) object wrapping value.

6.620.3 Field Documentation

6.620.3.1 `const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF`
[static]

Max Value for this Object's primitive type.

6.620.3.2 `const short decaf::lang::Short::MIN_VALUE = (short)0x8000` [static]

Max Value for this Object's primitive type.

6.620.3.3 `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Short.h`

6.621 decaf::internal::nio::ShortArrayBuffer Class Reference

#include <src/main/decaf/internal/nio/ShortArrayBuffer.h> Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

Public Member Functions

- **ShortArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **ShortArrayBuffer** (p. 2915) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ShortArrayBuffer** (short *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **ShortArrayBuffer** (p. 2915) object that wraps the given array.*
- **ShortArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset.*
- **ShortArrayBuffer** (const ShortArrayBuffer &other)
*Create a **ShortArrayBuffer** (p. 2915) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.*
- virtual ~**ShortArrayBuffer** ()
- virtual short * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the short array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual ShortBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only short buffer that shares this buffer's content.
- virtual ShortBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual ShortBuffer * **duplicate** ()
Creates a new short buffer that shares this buffer's content.
- virtual short **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual short **get** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible short array.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

- virtual ShortBuffer & **put** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual ShortBuffer & **put** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given doubles into this buffer at the given index.

- virtual ShortBuffer * **slice** () const

Creates a new ShortBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 870) as Read-Only.*

6.621.1 Constructor & Destructor Documentation

6.621.1.1 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **ShortArrayBuffer** (p. 2915) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.621.1.2 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **ShortArrayBuffer** (p. 2915) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.621.1.3 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 908) and start at the given offset. The capacity and limit of the new **ShortArrayBuffer** (p. 2915) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 908) to wrap

offset - the position that is this buffers start pos.

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.621.1.4 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const ShortArrayBuffer & other)

Create a **ShortArrayBuffer** (p. 2915) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 908) and when changes are made to that data it is reflected in both.

Parameters:

other - the **ShortArrayBuffer** (p. 2915) this one is to mirror.

6.621.1.5 virtual decaf::internal::nio::ShortArrayBuffer::~~ShortArrayBuffer () [virtual]

6.621.2 Member Function Documentation

6.621.2.1 virtual short* decaf::internal::nio::ShortArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the short array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2926).

6.621.2.2 `virtual std::size_t decaf::internal::nio::ShortArrayBuffer::arrayOffset
() throw (decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2926).

6.621.2.3 `virtual ShortBuffer* de-
caf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer ()
const [virtual]`

Creates a new, read-only short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 2926).

6.621.2.4 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 807) is copied

to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index `limit()` (p. 807) - 1 is copied to index $n = \text{limit}() - 1 - p$. The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this ShortBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2927).

6.621.2.5 virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate ()
[virtual]

Creates a new short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new short Buffer which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2927).

6.621.2.6 virtual short decaf::internal::nio::ShortArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the short is to be read

Returns:

the short that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ShortBuffer** (p. 2929).

6.621.2.7 `virtual short decaf::internal::nio::ShortArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the short at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements `decaf::nio::ShortBuffer` (p. 2929).

6.621.2.8 `virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible short array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::ShortBuffer` (p. 2929).

6.621.2.9 `virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements `decaf::nio::Buffer` (p. 806).

6.621.2.10 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2930).

6.621.2.11 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2930).

6.621.2.12 `virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **ByteBuffer** (p. 870) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.621.2.13 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice () const` [virtual]

Creates a new **ShortBuffer** whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ShortBuffer** which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2932).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ShortArrayBuffer.h`

6.622 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:.

#include <src/main/decaf/nio/ShortBuffer.h> Inheritance diagram for decaf::nio::ShortBuffer:

Public Member Functions

- virtual **~ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the short array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **ShortBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only short buffer that shares this buffer's content.
- virtual **ShortBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **ShortBuffer** * **duplicate** ()=0
Creates a new short buffer that shares this buffer's content.
- virtual short **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual short **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **ShortBuffer** & **get** (std::vector< short > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **ShortBuffer** & **get** (short *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible short array.
- **ShortBuffer** & **put** (**ShortBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the shorts remaining in the given source buffer into this buffer.

- **ShortBuffer** & **put** (const short *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers shorts into this buffer from the given source array.

- **ShortBuffer** & **put** (std::vector< short > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source shorts array into this buffer.

- virtual **ShortBuffer** & **put** (short value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

Writes the given shorts into this buffer at the current position, and then increments the position.

- virtual **ShortBuffer** & **put** (std::size_t index, short value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes the given shorts into this buffer at the given index.

- virtual **ShortBuffer** * **slice** () const =0

*Creates a new **ShortBuffer** (p. 2923) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ShortBuffer** &value) const

Compares this object with the specified object for order.

- virtual bool **equals** (const **ShortBuffer** &value) const

- virtual bool **operator==** (const **ShortBuffer** &value) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const **ShortBuffer** &value) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **ShortBuffer** * **allocate** (std::size_t capacity)

Allocates a new Double buffer.

- static **ShortBuffer** * **wrap** (short *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

*Wraps the passed buffer with a new **ShortBuffer** (p. 2923).*

- static **ShortBuffer** * **wrap** (std::vector< short > &buffer)

*Wraps the passed STL short Vector in a **ShortBuffer** (p. 2923).*

Protected Member Functions

- **ShortBuffer** (std::size_t capacity)

*Creates a **ShortBuffer** (p. 2923) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.622.1 Detailed Description

This class defines four categories of operations upon short buffers:.

- o Absolute and relative get and put methods that read and write single shorts;
- o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array; and
- o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer
- o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.622.2 Constructor & Destructor Documentation

6.622.2.1 decaf::nio::ShortBuffer::ShortBuffer (std::size_t capacity) [protected]

Creates a **ShortBuffer** (p. 2923) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 803) in doubles

6.622.2.2 virtual decaf::nio::ShortBuffer::~~ShortBuffer () [inline, virtual]

6.622.3 Member Function Documentation

6.622.3.1 static ShortBuffer* decaf::nio::ShortBuffer::allocate (std::size_t capacity) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in shorts

Returns:

the **ShortBuffer** (p. 2923) that was allocated, caller owns.

6.622.3.2 `virtual short* decaf::nio::ShortBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the short array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 803)

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 2917).

6.622.3.3 `virtual std::size_t decaf::nio::ShortBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException (p. 2685) if this **Buffer** (p. 803) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 2918).

6.622.3.4 `virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer () const [pure virtual]`

Creates a new, read-only short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only short buffer which the caller then owns.

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 2918).

6.622.3.5 virtual ShortBuffer& decaf::nio::ShortBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 807) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 807) - 1 is copied to index $n = \text{limit}()$ (p. 807) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **ShortBuffer** (p. 2923)

Exceptions:

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2918).

6.622.3.6 virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & value) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.622.3.7 virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate () [pure virtual]

Creates a new short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new short **Buffer** (p. 803) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2919).

6.622.3.8 virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & *value*)
const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.622.3.9 ShortBuffer& decaf::nio::ShortBuffer::get (short * *buffer*, std::size_t
offset, std::size_t *length*) throw (BufferUnderflowException,
lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if *length* > **remaining()** (p. 808), then no bytes are transferred and a **BufferUnderflowException** (p. 837) is thrown.

Otherwise, this method copies *length* shorts from this buffer into the given array, starting at the current position of this buffer and at the given *offset* in the array. The position of this buffer is then incremented by *length*.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than *length* shorts remaining in this buffer

NullPointerException if the passed buffer is null.

6.622.3.10 ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > *buffer*)
throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form *src.get(a)* behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call *buffer.resize(N)* before calling this get method.

Returns:

a reference to this **Buffer** (p. 803)

Exceptions:

BufferUnderflowException (p. 837) - If there are fewer than *length* shorts remaining in this buffer

6.622.3.11 `virtual short decaf::nio::ShortBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 803) where the short is to be read

Returns:

the short that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2919).

6.622.3.12 `virtual short decaf::nio::ShortBuffer::get () throw (BufferUnderflowException) [pure virtual]`

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the short at the current position

Exceptions:

BufferUnderflowException (p. 837) if there no more data to return

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2920).

6.622.3.13 `virtual bool decaf::nio::ShortBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible short array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2920).

6.622.3.14 `virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.622.3.15 `virtual bool decaf::nio::ShortBuffer::operator==(const ShortBuffer & value) const` [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.622.3.16 `virtual ShortBuffer& decaf::nio::ShortBuffer::put (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given shorts into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 803) to write the data

value - the shorts to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 2920).

6.622.3.17 `virtual ShortBuffer& decaf::nio::ShortBuffer::put (short value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters:

value - the shorts value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2921).

6.622.3.18 ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source shorts array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size())

Parameters:

buffer - The buffer whose contents are copied to this **ShortBuffer** (p. 2923)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.622.3.19 ShortBuffer& decaf::nio::ShortBuffer::put (const short * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers shorts into this buffer from the given source array. If there are more shorts to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 808), then no shorts are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which shorts are to be read

offset- The offset within the array of the first short to be read;

length - The number of shorts to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.622.3.20 `ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & src)
throw (BufferOverflowException, ReadOnlyBufferException,
lang::exceptions::IllegalArgumentException)`

This method transfers the shorts remaining in the given source buffer into this buffer. If there are more shorts remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 808), then no shorts are transferred and a **BufferOverflowException** (p. 834) is thrown.

Otherwise, this method copies `n = src.remaining()` shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters:

src - the buffer to take shorts from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. 834) - If there is insufficient space in this buffer for the remaining shorts in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2685) - If this buffer is read-only

6.622.3.21 `virtual ShortBuffer* decaf::nio::ShortBuffer::slice () const [pure
virtual]`

Creates a new **ShortBuffer** (p. 2923) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ShortBuffer** (p. 2923) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2921).

6.622.3.22 `virtual std::string decaf::nio::ShortBuffer::toString () const [virtual]`

Returns:

a `std::string` describing this object

6.622.3.23 `static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & buffer)` [static]

Wraps the passed STL short Vector in a **ShortBuffer** (p.2923). The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns:

a new **ShortBuffer** (p.2923) that is backed by *buffer*, caller owns.

6.622.3.24 `static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)` [static]

Wraps the passed buffer with a new **ShortBuffer** (p.2923). The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **ShortBuffer** (p.2923) that is backed by *buffer*, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ShortBuffer.h`

6.623 activemq::commands::ShutdownInfo Class Reference

#include <src/main/activemq/commands/ShutdownInfo.h> Inheritance diagram for activemq::commands::ShutdownInfo:

Public Member Functions

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ShutdownInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _SHUTDOWNINFO** = 11

Protected Member Functions

- **ShutdownInfo** (const **ShutdownInfo** &)
- **ShutdownInfo** & operator= (const **ShutdownInfo** &)

6.623.1 Constructor & Destructor Documentation

6.623.1.1 `activemq::commands::ShutdownInfo::ShutdownInfo (const ShutdownInfo &) [inline, protected]`

6.623.1.2 `activemq::commands::ShutdownInfo::ShutdownInfo ()`

6.623.1.3 `virtual activemq::commands::ShutdownInfo::~~ShutdownInfo () [virtual]`

6.623.2 Member Function Documentation

6.623.2.1 `virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.623.2.2 `virtual void activemq::commands::ShutdownInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.623.2.3 `virtual bool activemq::commands::ShutdownInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.623.2.4 `virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

6.623.2.5 **virtual bool activemq::commands::ShutdownInfo::isShutdownInfo ()**
const [inline, virtual]

Returns:

an answer of true to the **isShutdownInfo()** (p. 2936) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.623.2.6 **ShutdownInfo& activemq::commands::ShutdownInfo::operator= (const**
ShutdownInfo &) [inline, protected]

6.623.2.7 **virtual std::string activemq::commands::ShutdownInfo::toString () const**
[virtual]

Returns a string containing the information for this **DataSet** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.623.2.8 **virtual Pointer<Command> activemq::commands::ShutdownInfo::visit**
(activemq::state::CommandVisitor * *visitor*) throw (
exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1067).

6.623.3 Field Documentation

6.623.3.1 **const unsigned char activemq::commands::ShutdownInfo::ID_ -**
SHUTDOWNINFO = 11 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ShutdownInfo.h**

6.624 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2937).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.624.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2937). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.624.2 Constructor & Destructor Documentation

6.624.2.1 `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.624.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.624.3 Member Function Documentation

6.624.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.624.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.624.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 665).

6.624.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 666).

6.624.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 667).

6.624.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.624.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ShutdownInfoMarshaller.h**

6.625 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2941).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.625.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2941). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.625.2 Constructor & Destructor Documentation

6.625.2.1 `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.625.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.625.3 Member Function Documentation

6.625.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.625.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.625.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 672).

6.625.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 673).

6.625.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 674).

6.625.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.625.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ShutdownInfoMarshaller.h**

6.626 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2945).

#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.626.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2945). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.626.2 Constructor & Destructor Documentation

6.626.2.1 `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.626.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.626.3 Member Function Documentation

6.626.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.626.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.626.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.626.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.626.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.626.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.626.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**

6.627 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2949).

#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.627.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2949). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.627.2 Constructor & Destructor Documentation

6.627.2.1 `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.627.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.627.3 Member Function Documentation

6.627.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.627.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.627.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.627.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.627.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.627.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.627.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ShutdownInfoMarshaller.h**

6.628 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2953).

#include <src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.628.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2953). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.628.2 Constructor & Destructor Documentation

6.628.2.1 `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.628.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.628.3 Member Function Documentation

6.628.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.628.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.628.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 679).

6.628.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 680).

6.628.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 681).

6.628.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.628.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ShutdownInfoMarshaller.h**

6.629 decaf::security::SignatureException Class Reference

#include <src/main/decaf/security/SignatureException.h> Inheritance diagram for decaf::security::SignatureException:

Public Member Functions

- **SignatureException** () throw ()
Default Constructor.
- **SignatureException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **SignatureException** (const **SignatureException** &ex) throw ()
Copy Constructor.
- **SignatureException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SignatureException** (const std::exception *cause) throw ()
Constructor.
- **SignatureException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SignatureException** * clone () const
Clones this exception.
- virtual ~**SignatureException** () throw ()

6.629.1 Constructor & Destructor Documentation

6.629.1.1 decaf::security::SignatureException::SignatureException () throw () [inline]

Default Constructor.

6.629.1.2 decaf::security::SignatureException::SignatureException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.629.1.3 decaf::security::SignatureException::SignatureException (const SignatureException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.629.1.4 decaf::security::SignatureException::SignatureException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.629.1.5 decaf::security::SignatureException::SignatureException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.629.1.6 decaf::security::SignatureException::SignatureException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.629.1.7 virtual decaf::security::SignatureException::~~SignatureException ()
throw () [inline, virtual]

6.629.2 Member Function Documentation

6.629.2.1 virtual SignatureException* decaf::security::SignatureException::clone ()
const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1708).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SignatureException.h**

6.630 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 2050) in a human readable format.

#include <src/main/decaf/util/logging/SimpleFormatter.h> Inheritance diagram for decaf::util::logging::SimpleFormatter:

Public Member Functions

- **SimpleFormatter** ()
- virtual **~SimpleFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const
Format the given log record and return the formatted string.
- virtual std::string **formatMessage** (const **LogRecord** &record) const
Format the message string from a log record.
- virtual std::string **getHead** (const **Handler** *handler)
Return the header string for a set of formatted records.
- virtual std::string **getTail** (const **Handler** *handler)
Return the tail string for a set of formatted records.

6.630.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 2050) in a human readable format. The summary will typically be 1 or 2 lines.

6.630.2 Constructor & Destructor Documentation

6.630.2.1 decaf::util::logging::SimpleFormatter::SimpleFormatter () [inline]

6.630.2.2 virtual decaf::util::logging::SimpleFormatter::~~SimpleFormatter ()
[inline, virtual]

6.630.3 Member Function Documentation

6.630.3.1 virtual std::string decaf::util::logging::SimpleFormatter::format (const **LogRecord** & *record*) const [inline, virtual]

Format the given log record and return the formatted string.

Parameters:

record The Log Record to Format

Implements **decaf::util::logging::Formatter** (p. 1699).

6.630.3.2 virtual std::string decaf::util::logging::SimpleFormatter::formatMessage (const LogRecord & *record*) const [inline, virtual]

Format the message string from a log record.

Parameters:

record The Log Record to Format

Implements **decaf::util::logging::Formatter** (p. 1700).

References **decaf::util::logging::LogRecord::getMessage()**.

6.630.3.3 virtual std::string decaf::util::logging::SimpleFormatter::getHead (const Handler * *handler*) [inline, virtual]

Return the header string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

empty string

Implements **decaf::util::logging::Formatter** (p. 1700).

6.630.3.4 virtual std::string decaf::util::logging::SimpleFormatter::getTail (const Handler * *handler*) [inline, virtual]

Return the tail string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

empty string

Implements **decaf::util::logging::Formatter** (p. 1700).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleFormatter.h`

6.631 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

Public Member Functions

- **SimpleLogger** (const std::string &name)
Constructor.
- virtual ~**SimpleLogger** ()
Destructor.
- virtual void **mark** (const std::string &message)
Log a Mark Block Level Log.
- virtual void **debug** (const std::string &file, const int line, const std::string &message)
Log a Debug Level Log.
- virtual void **info** (const std::string &file, const int line, const std::string &message)
Log a Informational Level Log.
- virtual void **warn** (const std::string &file, const int line, const std::string &message)
Log a Warning Level Log.
- virtual void **error** (const std::string &file, const int line, const std::string &message)
Log a Error Level Log.
- virtual void **fatal** (const std::string &file, const int line, const std::string &message)
Log a Fatal Level Log.
- virtual void **log** (const std::string &message)
No-frills log.

6.631.1 Constructor & Destructor Documentation

6.631.1.1 decaf::util::logging::SimpleLogger::SimpleLogger (const std::string &name)

Constructor.

6.631.1.2 virtual decaf::util::logging::SimpleLogger::~~SimpleLogger () [virtual]

Destructor.

6.631.2 Member Function Documentation

6.631.2.1 virtual void decaf::util::logging::SimpleLogger::debug (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Debug Level Log.

6.631.2.2 virtual void decaf::util::logging::SimpleLogger::error (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Error Level Log.

6.631.2.3 virtual void decaf::util::logging::SimpleLogger::fatal (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Fatal Level Log.

6.631.2.4 virtual void decaf::util::logging::SimpleLogger::info (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Informational Level Log.

6.631.2.5 virtual void decaf::util::logging::SimpleLogger::log (const std::string & *message*) [virtual]

No-frills log.

6.631.2.6 virtual void decaf::util::logging::SimpleLogger::mark (const std::string & *message*) [virtual]

Log a Mark Block Level Log.

6.631.2.7 virtual void decaf::util::logging::SimpleLogger::warn (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Warning Level Log.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleLogger.h**

6.632 decaf::net::Socket Class Reference

#include <src/main/decaf/net/Socket.h> Inheritance diagram for decaf::net::Socket:

Public Types

- typedef apr_socket_t * **SocketHandle**
Define the SocketHandle type.
- typedef apr_sockaddr_t * **SocketAddress**
Define the SocketAddress type.

Public Member Functions

- virtual ~**Socket** ()
- virtual void **connect** (const char *host, int port)=0 throw (SocketException)
Connects to the specified destination.
- virtual bool **isConnected** () const =0
Indicates whether or not this socket is connected to a destination.
- virtual io::InputStream * **getInputStream** ()=0
Gets the InputStream for this socket.
- virtual io::OutputStream * **getOutputStream** ()=0
Gets the OutputStream for this socket.
- virtual int **getSoLinger** () const =0 throw (SocketException)
Gets the linger time.
- virtual void **setSoLinger** (int linger)=0 throw (SocketException)
Sets the linger time.
- virtual bool **getKeepAlive** () const =0 throw (SocketException)
Gets the keep alive flag.
- virtual void **setKeepAlive** (bool keepAlive)=0 throw (SocketException)
Enables/disables the keep alive flag.
- virtual int **getReceiveBufferSize** () const =0 throw (SocketException)
Gets the receive buffer size.
- virtual void **setReceiveBufferSize** (int size)=0 throw (SocketException)
Sets the receive buffer size.
- virtual bool **getReuseAddress** () const =0 throw (SocketException)

Gets the reuse address flag.

- virtual void **setReuseAddress** (bool reuse)=0 throw (SocketException)
Sets the reuse address flag.
- virtual int **getSendBufferSize** () const =0 throw (SocketException)
Gets the send buffer size.
- virtual void **setSendBufferSize** (int size)=0 throw (SocketException)
Sets the send buffer size.
- virtual int **getSoTimeout** () const =0 throw (SocketException)
Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout)=0 throw (SocketException)
Sets the timeout for socket operations.

6.632.1 Member Typedef Documentation

6.632.1.1 typedef apr_sockaddr_t* decaf::net::Socket::SocketAddress

Define the SocketAddress type.

6.632.1.2 typedef apr_socket_t* decaf::net::Socket::SocketHandle

Define the SocketHandle type.

6.632.2 Constructor & Destructor Documentation

6.632.2.1 virtual decaf::net::Socket::~~Socket () [inline, virtual]

6.632.3 Member Function Documentation

6.632.3.1 virtual void decaf::net::Socket::connect (const char * *host*, int *port*) throw (SocketException) [pure virtual]

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions:

IOException Thrown if a failure occurred in the connect.

Implemented in **decaf::net::BufferedSocket** (p. 819), and **decaf::net::TcpSocket** (p. 3154).

6.632.3.2 `virtual io::InputStream* decaf::net::Socket::getInputStream ()` [pure virtual]

Gets the InputStream for this socket.

Returns:

The InputStream for this socket. NULL if not connected.

Implemented in `decaf::net::BufferedSocket` (p. 819), and `decaf::net::TcpSocket` (p. 3155).

6.632.3.3 `virtual bool decaf::net::Socket::getKeepAlive () const throw (SocketException)` [pure virtual]

Gets the keep alive flag.

Returns:

True if keep alive is enabled.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in `decaf::net::BufferedSocket` (p. 819), and `decaf::net::TcpSocket` (p. 3155).

Referenced by `decaf::net::BufferedSocket::getKeepAlive()`.

6.632.3.4 `virtual io::OutputStream* decaf::net::Socket::getOutputStream ()` [pure virtual]

Gets the OutputStream for this socket.

Returns:

the OutputStream for this socket. NULL if not connected.

Implemented in `decaf::net::BufferedSocket` (p. 819), and `decaf::net::TcpSocket` (p. 3155).

6.632.3.5 `virtual int decaf::net::Socket::getReceiveBufferSize () const throw (SocketException)` [pure virtual]

Gets the receive buffer size.

Returns:

the receive buffer size in bytes.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in `decaf::net::BufferedSocket` (p. 820), and `decaf::net::TcpSocket` (p. 3155).

Referenced by `decaf::net::BufferedSocket::getReceiveBufferSize()`.

6.632.3.6 virtual bool decaf::net::Socket::getReuseAddress () const throw (SocketException) [pure virtual]

Gets the reuse address flag.

Returns:

True if the address can be reused.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 820), and **decaf::net::TcpSocket** (p. 3156).

Referenced by decaf::net::BufferedSocket::getReuseAddress().

6.632.3.7 virtual int decaf::net::Socket::getSendBufferSize () const throw (SocketException) [pure virtual]

Gets the send buffer size.

Returns:

the size in bytes of the send buffer.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 820), and **decaf::net::TcpSocket** (p. 3156).

Referenced by decaf::net::BufferedSocket::getSendBufferSize().

6.632.3.8 virtual int decaf::net::Socket::getSoLinger () const throw (SocketException) [pure virtual]

Gets the linger time.

Returns:

The linger time in seconds.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 820), and **decaf::net::TcpSocket** (p. 3156).

Referenced by decaf::net::BufferedSocket::getSoLinger().

6.632.3.9 virtual int decaf::net::Socket::getSoTimeout () const throw (SocketException) [pure virtual]

Gets the timeout for socket operations.

Returns:

The timeout in milliseconds for socket operations.

Exceptions:

SocketException (p. 2972) Thrown if unable to retrieve the information.

Implemented in **decaf::net::BufferedSocket** (p. 821), and **decaf::net::TcpSocket** (p. 3157).

Referenced by **decaf::net::BufferedSocket::getSoTimeout()**.

6.632.3.10 virtual bool decaf::net::Socket::isConnected () const [pure virtual]

Indicates whether or not this socket is connected to a destination.

Returns:

true if connected

Implemented in **decaf::net::BufferedSocket** (p. 821), and **decaf::net::TcpSocket** (p. 3157).

Referenced by **decaf::net::BufferedSocket::isConnected()**.

6.632.3.11 virtual void decaf::net::Socket::setKeepAlive (bool *keepAlive*) throw (SocketException) [pure virtual]

Enables/disables the keep alive flag.

Parameters:

keepAlive If true, enables the flag.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 821), and **decaf::net::TcpSocket** (p. 3157).

Referenced by **decaf::net::BufferedSocket::setKeepAlive()**.

6.632.3.12 virtual void decaf::net::Socket::setReceiveBufferSize (int *size*) throw (SocketException) [pure virtual]

Sets the receive buffer size.

Parameters:

size Number of bytes to set the receive buffer to.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 822), and **decaf::net::TcpSocket** (p. 3158).

Referenced by **decaf::net::BufferedSocket::setReceiveBufferSize()**.

6.632.3.13 `virtual void decaf::net::Socket::setReuseAddress (bool reuse) throw (SocketException)` [pure virtual]

Sets the reuse address flag.

Parameters:

reuse If true, sets the flag.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 822), and **decaf::net::TcpSocket** (p. 3158).

Referenced by `decaf::net::BufferedSocket::setReuseAddress()`.

6.632.3.14 `virtual void decaf::net::Socket::setSendBufferSize (int size) throw (SocketException)` [pure virtual]

Sets the send buffer size.

Parameters:

size The number of bytes to set the send buffer to.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 822), and **decaf::net::TcpSocket** (p. 3158).

Referenced by `decaf::net::BufferedSocket::setSendBufferSize()`.

6.632.3.15 `virtual void decaf::net::Socket::setSoLinger (int linger) throw (SocketException)` [pure virtual]

Sets the linger time.

Parameters:

linger The linger time in seconds. If 0, linger is off.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 822), and **decaf::net::TcpSocket** (p. 3158).

Referenced by `decaf::net::BufferedSocket::setSoLinger()`.

6.632.3.16 `virtual void decaf::net::Socket::setSoTimeout (int timeout) throw (SocketException)` [pure virtual]

Sets the timeout for socket operations.

Parameters:

timeout The timeout in milliseconds for socket operations.

Exceptions:

SocketException (p. 2972) Thrown if unable to set the information.

Implemented in **decaf::net::BufferedSocket** (p. 823), and **decaf::net::TcpSocket** (p. 3159).

Referenced by `decaf::net::BufferedSocket::setSoTimeout()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Socket.h`

6.633 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

```
#include <src/main/decaf/net/SocketError.h>
```

Static Public Member Functions

- static int **getErrorCode** ()
Gets the last error appropriate for the platform.
- static std::string **getErrorString** ()
Gets the string description for the last error.

6.633.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

6.633.2 Member Function Documentation

6.633.2.1 static int decaf::net::SocketError::getErrorCode () [static]

Gets the last error appropriate for the platform.

6.633.2.2 static std::string decaf::net::SocketError::getErrorString () [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketError.h**

6.634 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

#include <src/main/decaf/net/SocketException.h> Inheritance diagram for decaf::net::SocketException:

Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const lang::Exception &ex) throw ()
- **SocketException** (const SocketException &ex) throw ()
- **SocketException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketException** (const std::exception *cause) throw ()
Constructor.
- **SocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketException** * clone () const
Clones this exception.
- virtual ~**SocketException** () throw ()

6.634.1 Detailed Description

Exception for errors when manipulating sockets.

6.634.2 Constructor & Destructor Documentation

- 6.634.2.1 decaf::net::SocketException::SocketException () throw () [inline]
- 6.634.2.2 decaf::net::SocketException::SocketException (const lang::Exception &ex) throw () [inline]
- 6.634.2.3 decaf::net::SocketException::SocketException (const SocketException &ex) throw () [inline]
- 6.634.2.4 decaf::net::SocketException::SocketException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.634.2.5 decaf::net::SocketException::SocketException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.634.2.6 decaf::net::SocketException::SocketException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.634.2.7 virtual decaf::net::SocketException::~SocketException () throw () [inline, virtual]

6.634.3 Member Function Documentation

6.634.3.1 virtual SocketException* decaf::net::SocketException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1822).

Reimplemented in **decaf::net::BindException** (p. 723), **decaf::net::ConnectException** (p. 1130), **decaf::net::NoRouteToHostException** (p. 2419), and **decaf::net::PortUnreachableException** (p. 2524).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketException.h`

6.635 decaf::net::SocketFactory Class Reference

Socket (p. 2964) Factory implementation for use in Creating Sockets.

```
#include <src/main/decaf/net/SocketFactory.h>
```

Public Member Functions

- virtual `~SocketFactory()`

Static Public Member Functions

- static `Socket * createSocket (const std::string &uri, const util::Properties &properties) throw (SocketException)`

*Creates and returns a **Socket** (p. 2964) derived Object based on the values defined in the Properties Object that is passed in.*

6.635.1 Detailed Description

Socket (p. 2964) Factory implementation for use in Creating Sockets. Property Options:

Name Value

inputBufferSize size in bytes of the buffered input stream buffer. Defaults to 10000.

outputBufferSize size in bytes of the buffered output stream buffer. Defaults to 10000.

soLinger linger time for the socket (in microseconds). Defaults to 0.

soKeepAlive keep alive flag for the socket (true/false). Defaults to false.

soReceiveBufferSize The size of the socket receive buffer (in bytes). Defaults to 2MB.

soSendBufferSize The size of the socket send buffer (in bytes). Defaults to 2MB.

soTimeout The timeout of socket IO operations (in microseconds). Defaults to 10000

See also:

`Socket` (p. 2964)

6.635.2 Constructor & Destructor Documentation

6.635.2.1 `virtual decaf::net::SocketFactory::~~SocketFactory () [inline, virtual]`

6.635.3 Member Function Documentation

6.635.3.1 `static Socket* decaf::net::SocketFactory::createSocket (const std::string & uri, const util::Properties & properties) throw (SocketException) [static]`

Creates and returns a **Socket** (p. 2964) derived Object based on the values defined in the Properties Object that is passed in.

Parameters:

- uri* the **URI** (p. 3308) for the **Socket** (p. 2964) Connection.
- properties* A Properties object that contains configuration details.

Exceptions:

SocketException.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketFactory.h`

6.636 decaf::net::SocketInputStream Class Reference

Input stream for performing reads on a socket.

#include <src/main/decaf/net/SocketInputStream.h> Inheritance diagram for decaf::net::SocketInputStream:

Public Member Functions

- **SocketInputStream** (**Socket::SocketHandle** socket)
Constructor.
- virtual **~SocketInputStream** ()
Destructor.
- virtual std::size_t **available** () const throw (io::IOException)
Returns the number of bytes available on the socket to be read right now.
- virtual int **read** () throw (io::IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (io::IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (decaf::io::IOException)
Close - does nothing.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Not supported.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (io::IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.636.1 Detailed Description

Input stream for performing reads on a socket. This class will only work properly for blocking sockets.

6.636.2 Constructor & Destructor Documentation

6.636.2.1 decaf::net::SocketInputStream::SocketInputStream (Socket::SocketHandle *socket*)

Constructor.

Parameters:

socket the socket handle.

6.636.2.2 virtual decaf::net::SocketInputStream::~~SocketInputStream () [virtual]

Destructor.

6.636.3 Member Function Documentation

6.636.3.1 `virtual std::size_t decaf::net::SocketInputStream::available () const throw (io::IOException) [virtual]`

Returns the number of bytes available on the socket to be read right now.

Returns:

The number of bytes currently available to be read on the socket.

Implements **decaf::io::InputStream** (p. 1741).

6.636.3.2 `virtual void decaf::net::SocketInputStream::close () throw (decaf::io::IOException) [virtual]`

Close - does nothing. It is the responsibility of the owner of the socket object to close it.

Exceptions:

IOException

Implements **decaf::io::Closeable** (p. 1019).

6.636.3.3 `virtual void decaf::net::SocketInputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3123).

6.636.3.4 `virtual void decaf::net::SocketInputStream::mark (int readLimit DECAF_UNUSED) [inline, virtual]`

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1741).

6.636.3.5 `virtual bool decaf::net::SocketInputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1741).

6.636.3.6 **virtual void decaf::net::SocketInputStream::notify ()**
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3124).

6.636.3.7 **virtual void decaf::net::SocketInputStream::notifyAll**
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

6.636.3.8 **virtual int decaf::net::SocketInputStream::read (unsigned char * *buffer*,**
std::size_t *offset*, std::size_t *bufferSize*) throw (io::IOException,
lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer. If the desired amount of data is not currently available, this operation will block until the appropriate amount of data is available.

Parameters:

buffer (out) the target buffer

offset the position in the buffer to start from.

bufferSize the size of the output buffer.

Returns:

the number of bytes read. or -1 if EOF

Exceptions:

IOException if an error occurs.

Implements **decaf::io::InputStream** (p. 1742).

6.636.3.9 `virtual int decaf::net::SocketInputStream::read () throw (io::IOException) [virtual]`

Reads a single byte from the buffer. If no data is available, blocks until there is.

Returns:

The next byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1742).

6.636.3.10 `virtual void decaf::net::SocketInputStream::reset () throw (io::IOException) [inline, virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an IOException might be thrown. * If such an IOException is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an IOException. * If an IOException is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException

Implements **decaf::io::InputStream** (p. 1742).

6.636.3.11 `virtual std::size_t decaf::net::SocketInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported.

Exceptions:

an UnsupportedOperationException.

Implements **decaf::io::InputStream** (p. 1743).

6.636.3.12 **virtual bool decaf::net::SocketInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException)** [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.636.3.13 **virtual void decaf::net::SocketInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException)** [inline, virtual]

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.636.3.14 **virtual void decaf::net::SocketInputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)** [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.636.3.15 `virtual void decaf::net::SocketInputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.636.3.16 `virtual void decaf::net::SocketInputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketInputStream.h`

6.637 decaf::net::SocketOutputStream Class Reference

Output stream for performing write operations on a socket.

#include <src/main/decaf/net/SocketOutputStream.h> Inheritance diagram for decaf::net::SocketOutputStream:

Public Member Functions

- **SocketOutputStream** (**Socket::SocketHandle** socket)
Constructor.
- virtual **~SocketOutputStream** ()
- virtual void **write** (unsigned char c) throw (io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (io::IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (io::IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (io::IOException)
Flush - does nothing.
- virtual void **close** () throw (decaf::io::IOException)
Close - does nothing.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.637.1 Detailed Description

Output stream for performing write operations on a socket.

6.637.2 Constructor & Destructor Documentation

6.637.2.1 decaf::net::SocketOutputStream::SocketOutputStream (Socket::SocketHandle *socket*)

Constructor.

Parameters:

socket the socket handle.

6.637.2.2 virtual decaf::net::SocketOutputStream::~~SocketOutputStream () [virtual]

6.637.3 Member Function Documentation

6.637.3.1 virtual void decaf::net::SocketOutputStream::close () throw (decaf::io::IOException) [virtual]

Close - does nothing. It is the responsibility of the owner of the socket object to close it.

Exceptions:

IOException

Implements decaf::io::Closeable (p. 1019).

6.637.3.2 virtual void decaf::net::SocketOutputStream::flush () throw (io::IOException) [inline, virtual]

Flush - does nothing.

Exceptions:

IOException

Implements **decaf::io::OutputStream** (p. 2470).

6.637.3.3 `virtual void decaf::net::SocketOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3123).

6.637.3.4 `virtual void decaf::net::SocketOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3124).

6.637.3.5 `virtual void decaf::net::SocketOutputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

6.637.3.6 virtual bool decaf::net::SocketOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 3126).

6.637.3.7 virtual void decaf::net::SocketOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements decaf::util::concurrent::Synchronizable (p. 3127).

6.637.3.8 virtual void decaf::net::SocketOutputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.637.3.9 **virtual void decaf::net::SocketOutputStream::wait (long long *milliseconds*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)** [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.637.3.10 **virtual void decaf::net::SocketOutputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)** [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

6.637.3.11 **virtual void decaf::net::SocketOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (io::IOException, lang::exceptions::NullPointerException)** [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 2471).

6.637.3.12 `virtual void decaf::net::SocketOutputStream::write (const std::vector< unsigned char > & buffer) throw (io::IOException) [virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

6.637.3.13 `virtual void decaf::net::SocketOutputStream::write (unsigned char c) throw (io::IOException) [virtual]`

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketOutputStream.h`

6.638 decaf::net::SocketTimeoutException Class Reference

#include <src/main/decaf/net/SocketTimeoutException.h> Inheritance diagram for decaf::net::SocketTimeoutException:

Public Member Functions

- **SocketTimeoutException** () throw ()
Default Constructor.
- **SocketTimeoutException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()
Copy Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketTimeoutException** (const std::exception *cause) throw ()
Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketTimeoutException** * **clone** () const
Clones this exception.
- virtual ~**SocketTimeoutException** () throw ()

6.638.1 Constructor & Destructor Documentation

6.638.1.1 decaf::net::SocketTimeoutException::SocketTimeoutException () throw () [inline]

Default Constructor.

6.638.1.2 decaf::net::SocketTimeoutException::SocketTimeoutException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.638.1.3 decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.638.1.4 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.638.1.5 decaf::net::SocketTimeoutException::SocketTimeoutException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.638.1.6 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.638.1.7 `virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException
() throw () [inline, virtual]`

6.638.2 Member Function Documentation

6.638.2.1 `virtual SocketTimeoutException* de-
caf::net::SocketTimeoutException::clone () const [inline,
virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p.1807).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketTimeoutException.h`

6.639 activemq::commands::BrokerError::StackTraceElement Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

Data Fields

- `std::string` **ClassName**
- `std::string` **FileName**
- `std::string` **MethodName**
- `int` **LineNumber**

6.639.1 Field Documentation

6.639.1.1 `std::string` `activemq::commands::BrokerError::StackTraceElement::ClassName`

6.639.1.2 `std::string` `activemq::commands::BrokerError::StackTraceElement::FileName`

6.639.1.3 `int` `activemq::commands::BrokerError::StackTraceElement::LineNumber`

6.639.1.4 `std::string` `activemq::commands::BrokerError::StackTraceElement::MethodName`

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

6.640 decaf::internal::io::StandardErrorOutputStream Class Reference

Wrapper Around the Standard error Output facility on the current platform.

#include <src/main/decaf/internal/io/StandardErrorOutputStream.h> Inheritance diagram for decaf::internal::io::StandardErrorOutputStream:

Public Member Functions

- **StandardErrorOutputStream** ()

Default Constructor.

- virtual ~**StandardErrorOutputStream** ()

- virtual void **write** (unsigned char c) throw (decaf::io::IOException)

Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (decaf::io::IOException)

Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException)

Writes an array of bytes to the output stream.

- virtual void **flush** () throw (decaf::io::IOException)

Invokes flush on the target output stream.

- virtual void **close** () throw (decaf::io::IOException)

Invokes close on the target output stream.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.640.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform. This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

6.640.2 Constructor & Destructor Documentation

6.640.2.1 decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ()

Default Constructor.

6.640.2.2 virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream () [virtual]

6.640.3 Member Function Documentation

6.640.3.1 virtual void decaf::internal::io::StandardErrorOutputStream::close () throw (decaf::io::IOException) [inline, virtual]

Invokes close on the target output stream. throws IOException if an error occurs

Implements **decaf::io::Closeable** (p.1019).

6.640.3.2 virtual void decaf::internal::io::StandardErrorOutputStream::flush () throw (decaf::io::IOException) [virtual]

Invokes flush on the target output stream. throws **decaf::io::IOException** (p.1820) if an error occurs

Implements **decaf::io::OutputStream** (p.2470).

6.640.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::lock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 3123).

6.640.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::notify
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3124).

6.640.3.5 virtual void decaf::internal::io::StandardErrorOutputStream::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3125).

6.640.3.6 virtual bool decaf::internal::io::StandardErrorOutputStream::tryLock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.640.3.7 virtual void decaf::internal::io::StandardErrorOutputStream::unlock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.640.3.8 virtual void decaf::internal::io::StandardErrorOutputStream::wait
(long long *millisecs*, int *nanos*) throw (de-
caf::lang::exceptions::RuntimeException, de-
caf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.640.3.9 virtual void decaf::internal::io::StandardErrorOutputStream::wait (long long *millisecs*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements decaf::util::concurrent::Synchronizable (p. 3130).

6.640.3.10 virtual void decaf::internal::io::StandardErrorOutputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements decaf::util::concurrent::Synchronizable (p. 3131).

6.640.3.11 virtual void decaf::internal::io::StandardErrorOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (decaf::io::IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset The position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

decaf::io::IOException (p. 1820) thrown if an error occurs.

NullPointerException if buffer is null.

Implements **decaf::io::OutputStream** (p. 2471).

6.640.3.12 virtual void decaf::internal::io::StandardErrorOutputStream::write
(const std::vector< unsigned char > & *buffer*) throw (
decaf::io::IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

6.640.3.13 virtual void decaf::internal::io::StandardErrorOutputStream::write
(unsigned char *c*) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardErrorOutputStream.h**

6.641 decaf::internal::io::StandardInputStream Class Reference

#include <src/main/decaf/internal/io/StandardInputStream.h> Inheritance diagram for decaf::internal::io::StandardInputStream:

Public Member Functions

- **StandardInputStream** ()
- virtual **~StandardInputStream** ()
- virtual std::size_t **available** () const throw (decaf::io::IOException)
Indicates the number of bytes available.
- virtual int **read** () throw (decaf::io::IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (decaf::io::IOException)
Closes the target input stream.
- virtual std::size_t **skip** (std::size_t num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (decaf::io::IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.

Protected Member Functions

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.641.1 Constructor & Destructor Documentation

6.641.1.1 decaf::internal::io::StandardInputStream::StandardInputStream ()

6.641.1.2 virtual decaf::internal::io::StandardInputStream::~~StandardInputStream () [virtual]

6.641.2 Member Function Documentation

6.641.2.1 virtual std::size_t decaf::internal::io::StandardInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

Returns:

The number of bytes until the end of the internal buffer.

Implements **decaf::io::InputStream** (p. 1741).

6.641.2.2 `virtual void decaf::internal::io::StandardInputStream::close () throw (decaf::io::IOException)` [inline, virtual]

Closes the target input stream.

Exceptions:

IOException thrown if an error occurs.

Implements `decaf::io::Closeable` (p. 1019).

6.641.2.3 `virtual void decaf::internal::io::StandardInputStream::lock () throw (decaf::lang::exceptions::RuntimeException)` [inline, protected, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.641.2.4 `virtual void decaf::internal::io::StandardInputStream::mark (int readLimit DECAF_UNUSED)` [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1741).

6.641.2.5 `virtual bool decaf::internal::io::StandardInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1741).

6.641.2.6 virtual void decaf::internal::io::StandardInputStream::notify
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 protected, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3124).

6.641.2.7 virtual void decaf::internal::io::StandardInputStream::notifyAll
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 protected, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 3125).

6.641.2.8 virtual int decaf::internal::io::StandardInputStream::read (unsigned
 char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (decaf::io::IOException,
 decaf::lang::exceptions::NullPointerException)
 [virtual]

Reads an array of bytes from the buffer.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read.

Exceptions:

IOException thrown if an error occurs.

NullPointerException if buffer is null.

Implements decaf::io::InputStream (p. 1742).

6.641.2.9 `virtual int decaf::internal::io::StandardInputStream::read () throw (decaf::io::IOException) [virtual]`

Reads a single byte from the buffer.

Returns:

The next byte.

Exceptions:

IOException thrown if an error occurs.

Implements `decaf::io::InputStream` (p. 1742).

6.641.2.10 `virtual void decaf::internal::io::StandardInputStream::reset () throw (decaf::io::IOException) [virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an `IOException` might be thrown. * If such an `IOException` is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method `markSupported` returns false, then: * The call to reset may throw an `IOException`. * If an `IOException` is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException

Implements `decaf::io::InputStream` (p. 1742).

6.641.2.11 `virtual std::size_t decaf::internal::io::StandardInputStream::skip (std::size_t num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of `InputStream` creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException if an error occurs

UnsupportedOperationException If skip is not supported.

Implements **decaf::io::InputStream** (p. 1743).

6.641.2.12 virtual bool decaf::internal::io::StandardInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.641.2.13 virtual void decaf::internal::io::StandardInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.641.2.14 virtual void decaf::internal::io::StandardInputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.641.2.15 `virtual void decaf::internal::io::StandardInputStream::wait (long long milliseconds) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.641.2.16 `virtual void decaf::internal::io::StandardInputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardInputStream.h`

6.642 decaf::internal::io::StandardOutputStream Class Reference

#include <src/main/decaf/internal/io/StandardOutputStream.h> Inheritance diagram for decaf::internal::io::StandardOutputStream:

Public Member Functions

- **StandardOutputStream** ()
- virtual **~StandardOutputStream** ()
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (decaf::io::IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (decaf::io::IOException)
Invokes flush on the target output stream.
- virtual void **close** () throw (decaf::io::IOException)
Invokes close on the target output stream.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.642.1 Constructor & Destructor Documentation

6.642.1.1 decaf::internal::io::StandardOutputStream::StandardOutputStream ()

6.642.1.2 virtual decaf::internal::io::StandardOutputStream::~~StandardOutputStream () [virtual]

6.642.2 Member Function Documentation

6.642.2.1 virtual void decaf::internal::io::StandardOutputStream::close () throw (decaf::io::IOException) [inline, virtual]

Invokes close on the target output stream. throws IOException if an error occurs

Implements **decaf::io::Closeable** (p. 1019).

6.642.2.2 virtual void decaf::internal::io::StandardOutputStream::flush () throw (decaf::io::IOException) [virtual]

Invokes flush on the target output stream. throws IOException if an error occurs

Implements **decaf::io::OutputStream** (p. 2470).

6.642.2.3 virtual void decaf::internal::io::StandardOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3123).

6.642.2.4 `virtual void decaf::internal::io::StandardOutputStream::notify
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

6.642.2.5 `virtual void decaf::internal::io::StandardOutputStream::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3125).

6.642.2.6 `virtual bool decaf::internal::io::StandardOutputStream::tryLock () throw
(decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3126).

6.642.2.7 `virtual void decaf::internal::io::StandardOutputStream::unlock () throw (
decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.642.2.8 virtual void decaf::internal::io::StandardOutputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.642.2.9 virtual void decaf::internal::io::StandardOutputStream::wait (long long *millisecs*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.642.2.10 `virtual void decaf::internal::io::StandardOutputStream::wait
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

6.642.2.11 `virtual void decaf::internal::io::StandardOutputStream::write (const
unsigned char * buffer, std::size_t offset, std::size_t len) throw
(decaf::io::IOException, lang::exceptions::NullPointerException)
[virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset, the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException thrown if an error occurs.

NullPointerException if buffer is null.

Implements **decaf::io::OutputStream** (p. 2471).

6.642.2.12 `virtual void decaf::internal::io::StandardOutputStream::write (const
std::vector< unsigned char > & buffer) throw (decaf::io::IOException)
[virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

6.642.2.13 virtual void decaf::internal::io::StandardOutputStream::write (unsigned char *c*) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2471).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

6.643 cms::Startable Class Reference

Interface for a class that implements the start method.

#include <src/main/cms/Startable.h> Inheritance diagram for cms::Startable:

Public Member Functions

- virtual ~**Startable** ()
- virtual void **start** ()=0 throw (CMSEException)
Starts the service.

6.643.1 Detailed Description

Interface for a class that implements the start method. An object that implements the **Startable** (p. 3014) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since:

1.0

6.643.2 Constructor & Destructor Documentation

6.643.2.1 virtual cms::Startable::~~Startable () [inline, virtual]

6.643.3 Member Function Documentation

6.643.3.1 virtual void cms::Startable::start () throw (CMSEException) [pure virtual]

Starts the service.

Exceptions:

CMSEException (p. 1031) if an internal error occurs while starting.

Implemented in **activemq::core::ActiveMQConnection** (p. 242).

The documentation for this class was generated from the following file:

- src/main/cms/**Startable.h**

6.644 decaf::lang::STATIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.645 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Public Member Functions

- **StaticInitializer** ()
- virtual **~StaticInitializer** ()

Static Public Attributes

- static std::string **destOptions** [NUM_OPTIONS]
- static std::string **uriParams** [NUM_PARAMS]
- static std::map< std::string, **DestinationOption** > **destOptionMap**
- static std::map< std::string, **URIParam** > **uriParamsMap**

6.645.1 Constructor & Destructor Documentation

6.645.1.1 **activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer** ()

6.645.1.2 virtual **activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer** () [inline, virtual]

6.645.2 Field Documentation

6.645.2.1 std::map<std::string, **DestinationOption**> **activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap** [static]

6.645.2.2 std::string **activemq::core::ActiveMQConstants::StaticInitializer::destOptions**[NUM_OPTIONS] [static]

6.645.2.3 std::string **activemq::core::ActiveMQConstants::StaticInitializer::uriParams**[NUM_PARAMS] [static]

6.645.2.4 std::map<std::string, **URIParam**> **activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

6.646 decaf::util::StlList< E > Class Template Reference

List (p.1984) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

#include <src/main/decaf/util/StlList.h> Inheritance diagram for decaf::util::StlList< E >:

Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

Public Member Functions

- **StlList** ()
Default constructor - does nothing.
- **StlList** (const **StlList** &source)
Copy constructor - copies the content of the given set into this one.
- **StlList** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual ~**StlList** ()
- virtual bool **equals** (const **StlList** &source) const
- virtual **Iterator**< E > * **iterator** ()
Returns:
an iterator over a set of elements of type T.
- virtual **Iterator**< E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()
Returns:
a list iterator over the elements in this list (in proper sequence).
- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Parameters:
index index of first element to be returned from the list iterator (by a call to the next method).
Returns:
a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions:

IndexOutOfBoundsException if the index is out of range ($index < 0 \parallel index > size()$ (p. 1062))

- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const throw (default::lang::exceptions::IndexOutOfBoundsException)
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1833) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

- virtual bool **contains** (const E &value) const throw (lang::Exception)

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p. 1054).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

- virtual std::size_t **indexOf** (const E &value) throw (default::lang::exceptions::NoSuchElementException)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index i such that $get(i) == value$, or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the first occurrence of the specified element in this list,

Exceptions:

NoSuchElementException if value is not in the list

- virtual std::size_t **lastIndexOf** (const E &value) throw (default::lang::exceptions::NoSuchElementException)

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that $get(i) == value$ or -1 if there is no such index.

Parameters:*value* - element to search for**Returns:***the index of the last occurrence of the specified element in this list.***Exceptions:****NoSuchElementException** if *value* is not in the list

- virtual bool **isEmpty** () const

*Returns true if this collection contains no elements.**This implementation returns **size()** (p. 1062) == 0.***Returns:***true if the size method return 0.*

- virtual std::size_t **size** () const

*Returns the number of elements in this collection.**If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.***Returns:***the number of elements in this collection*

- virtual E **get** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

*Gets the element contained at position passed.***Parameters:***index* - position to get**Returns:***value at index*

- virtual E **set** (std::size_t index, const E &element) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

*Replaces the element at the specified position in this list with the specified element.***Parameters:***index* - index of the element to replace*element* - element to be stored at the specified position**Returns:***the element previously at the specified position***Exceptions:****IndexOutOfBoundsException** - if the *index* is greater than *size*

- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

*Returns true if this collection changed as a result of the call.**(Returns false if this collection does not permit duplicates and already contains the specified element.)**Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1054) classes should clearly specify in their documentation any restrictions on what elements may be added.*

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

- virtual void **add** (std::size_t index, const E &element) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException)

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters:

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

- virtual bool **addAll** (std::size_t index, const Collection< E > &source) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters:

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 1054) containing elements to be added to this list

Returns:

true if this list changed as a result of the call

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an *UnsupportedOperationException* if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this *Collection* (p. 1054).

- virtual E **remove** (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters:

index - the index of the element to be removed

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

6.646.1 Detailed Description

```
template<typename E> class decaf::util::StlList< E >
```

List (p.1984) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

6.646.2 Constructor & Destructor Documentation

6.646.2.1 template<typename E> decaf::util::StlList< E >::StlList () [inline]

Default constructor - does nothing.

6.646.2.2 template<typename E> decaf::util::StlList< E >::StlList (const StlList< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.646.2.3 `template<typename E> decaf::util::StlList< E >::StlList (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.646.2.4 `template<typename E> virtual decaf::util::StlList< E >::~StlList () [inline, virtual]`

6.646.3 Member Function Documentation

6.646.3.1 `template<typename E> virtual void decaf::util::StlList< E >::add (std::size_t index, const E & element) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters:

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements `decaf::util::List< E >` (p.1985).

6.646.3.2 `template<typename E> virtual bool decaf::util::StlList< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1054) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1056).

Referenced by **decaf::util::StlList< Pointer< BackupTransport > >::addAll()**.

```
6.646.3.3  template<typename E> virtual bool decaf::util::StlList< E
>::addAll (std::size_t index, const Collection< E > & source)
throw ( decaf::lang::exceptions::UnsupportedOperationException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters:

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 1054) containing elements to be added to this list

Returns:

true if this list changed as a result of the call

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 1985).

6.646.3.4 `template<typename E> virtual void decaf::util::StlList< E >::clear ()
throw (lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.1833) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from `decaf::util::AbstractCollection< E >` (p.151).

6.646.3.5 `template<typename E> virtual bool decaf::util::StlList< E >::contains
(const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p.1054).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

Reimplemented from `decaf::util::AbstractCollection< E >` (p.152).

6.646.3.6 `template<typename E> virtual void decaf::util::StlList< E >::copy (const
StlList< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlList< Pointer< BackupTransport > >::StlList()`.

6.646.3.7 `template<typename E> virtual bool decaf::util::StlList< E >::equals
(const StlList< E > & source) const [inline, virtual]`

6.646.3.8 `template<typename E> virtual E decaf::util::StlList< E >::get (std::size_t
index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException
) [inline, virtual]`

Gets the element contained at position passed.

Parameters:

index - position to get

Returns:

value at index

Implements **decaf::util::List< E >** (p.1986).

6.646.3.9 `template<typename E> virtual std::size_t decaf::util::StlList< E >::indexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the first occurrence of the specified element in this list,

Exceptions:

NoSuchElementException if value is not in the list

Implements **decaf::util::List< E >** (p.1986).

6.646.3.10 `template<typename E> virtual bool decaf::util::StlList< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns `size() (p.1062) == 0`.

Returns:

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p.153).

6.646.3.11 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p.1830).

6.646.3.12 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () [inline, virtual]`

Returns:

an iterator over a set of elements of type T.

Implements `decaf::lang::Iterable< E >` (p. 1830).

6.646.3.13 `template<typename E> virtual std::size_t decaf::util::StlList< E >::lastIndexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the last occurrence of the specified element in this list.

Exceptions:

NoSuchElementException if *value* is not in the list

Implements `decaf::util::List< E >` (p. 1987).

6.646.3.14 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Implements `decaf::util::List< E >` (p. 1987).

6.646.3.15 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Parameters:

index index of first element to be returned from the list iterator (by a call to the next method).

Returns:

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions:

IndexOutOfBoundsException if the index is out of range (index < 0 || index > size() (p. 1062))

Implements **decaf::util::List< E >** (p. 1988).

6.646.3.16 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 1988).

6.646.3.17 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () [inline, virtual]`

Returns:

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 1988).

6.646.3.18 `template<typename E> virtual E decaf::util::StlList< E >::remove (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters:

index - the index of the element to be removed

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 1989).

6.646.3.19 `template<typename E> virtual bool decaf::util::StlList< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1054).

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 155).

```
6.646.3.20  template<typename E> virtual E decaf::util::StlList< E
              >::set (std::size_t index, const E & element) throw (
              decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
              virtual]
```

Replaces the element at the specified position in this list with the specified element.

Parameters:

index - index of the element to replace

element - element to be stored at the specified position

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

Implements `decaf::util::List< E >` (p. 1989).

```
6.646.3.21  template<typename E> virtual std::size_t decaf::util::StlList< E >::size
              () const [inline, virtual]
```

Returns the number of elements in this collection.

If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

Returns:

the number of elements in this collection

Implements **decaf::util::Collection**< E > (p. 1062).

Referenced by decaf::util::StlList< Pointer< BackupTransport > >::add(), decaf::util::StlList< Pointer< BackupTransport > >::addAll(), decaf::util::StlList< Pointer< BackupTransport > >::get(), decaf::util::StlList< Pointer< BackupTransport > >::lastIndexOf(), decaf::util::StlList< Pointer< BackupTransport > >::listIterator(), decaf::util::StlList< Pointer< BackupTransport > >::remove(), and decaf::util::StlList< Pointer< BackupTransport > >::set().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlList.h**

6.647 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

#include <src/main/decaf/util/StlMap.h> Inheritance diagram for `decaf::util::StlMap< K, V, COMPARATOR >`:

Public Member Functions

- **StlMap** ()
Default constructor - does nothing.
- **StlMap** (const **StlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **StlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual ~**StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
Parameters:
source - **Map** (p. 2094) to compare to this one.
Returns:
*true if the **Map** (p. 2094) passed is equal in value to this one.*
- virtual void **copy** (const **StlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
Copies the content of the source map into this map.
Erases all existing data in this map.
Parameters:
source The source object to copy from.
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
Exceptions:
UnsupportedOperationException if this map is unmodifiable.
- virtual bool **containsKey** (const K &key) const
Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

- virtual bool **containsValue** (const V &value) const

Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const

Returns:

*if the **Map** (p. 2094) contains any element or not, TRUE or FALSE*

- virtual std::size_t **size** () const

Returns:

The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 2094).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2094).*

- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 2094).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 2094).*

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

Sets the value for the specified key.

Parameters:

key The target key.
value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

Stores a copy of the Mappings contained in the other Map (p. 2094) in this one.

Parameters:

other A **Map** (p. 2094) instance whose elements are to all be inserted in this **Map** (p. 2094).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 2094).
UnsupportedOperationException if this map is unmodifiable.

- virtual std::vector< K > **keySet** () const

Returns a Set (p. 2905) view of the mappings contained in this map.

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1833), **Set.remove** (p. 155), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.*

Returns:

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

Returns:

the entire set of values in this map as a std::vector.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.647.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::StlMap< K, V, COMPARATOR >
```

Map (p. 2094) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

Since:

1.0

6.647.2 Constructor & Destructor Documentation

6.647.2.1

```
template<typename K, typename V, typename COMPARATOR =
std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap ()
[inline]
```

Default constructor - does nothing.

6.647.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const StlMap< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.647.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const Map< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.647.2.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::StlMap< K, V, COMPARATOR >::~StlMap () [inline, virtual]`

6.647.3 Member Function Documentation

6.647.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear () throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all keys and values from this map.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2095).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.647.3.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsKey (const K & key) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2096).

Referenced by **decaf::util::StlMap< std::string, cms::Topic * >::equals()**.

6.647.3.3 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::containsValue (const V & value) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2097).

6.647.3.4 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::copy (const Map< K, V, COMPARATOR > & source) [inline,
virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters:

source The source object to copy from.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2098).

6.647.3.5 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::copy (const StlMap< K, V, COMPARATOR > & source) [inline,
virtual]`

Referenced by **decaf::util::StlMap< std::string, cms::Topic * >::StlMap()**.

6.647.3.6 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::equals (const Map< K, V, COMPARATOR > & source) const
[inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters:

source - **Map** (p. 2094) to compare to this one.

Returns:

true if the **Map** (p. 2094) passed is equal in value to this one.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2098).

6.647.3.7 `template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARTOR
>::equals (const StlMap< K, V, COMPARTOR > & source) const
[inline, virtual]`

6.647.3.8 `template<typename K, typename V, typename COMPARTOR
= std::less<K>> virtual const V& decaf::util::StlMap< K,
V, COMPARTOR >::get (const K & key) const throw (
lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2094).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2094).

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2098).

6.647.3.9 `template<typename K, typename V, typename COMPARTOR
= std::less<K>> virtual V& decaf::util::StlMap< K,
V, COMPARTOR >::get (const K & key) throw (
lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2094).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 2094).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2099).

```
6.647.3.10  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
            >::isEmpty () const [inline, virtual]
```

Returns:

if the **Map** (p. 2094) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2100).

```
6.647.3.11  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual std::vector<K> decaf::util::StlMap< K, V,
            COMPARATOR >::keySet () const [inline, virtual]
```

Returns a **Set** (p. 2905) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1833), **Set.remove** (p. 155), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns:

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2101).

```
6.647.3.12  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
            >::lock () throw ( decaf::lang::exceptions::RuntimeException ) [inline,
            virtual]
```

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3123).

6.647.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3124).

6.647.3.14 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

6.647.3.15 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::put (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Sets the value for the specified key.

Parameters:

key The target key.

value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2102).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::putAll()`.

6.647.3.16 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 2094) in this one.

Parameters:

other A **Map** (p. 2094) instance whose elements are to all be inserted in this **Map** (p. 2094).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2102).

6.647.3.17 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const StlMap< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.647.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove (const K & key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 2094).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2103).

6.647.3.19 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::StlMap< K, V, COMPARATOR >::size () const [inline, virtual]`

Returns:

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2104).

6.647.3.20 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3126).

6.647.3.21 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3127).

6.647.3.22 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<V> decaf::util::StlMap< K, V, COMPARATOR >::values () const [inline, virtual]`

Returns:

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2105).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::values()`.

```

6.647.3.23  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap< K,
            V, COMPARATOR >::wait (long long millisecs, int
            nanos) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalArgumentException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE
nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]
RuntimeException if an error occurs while waiting on the object.
InterruptedException if the wait is interrupted before it completes.
IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

```

6.647.3.24  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap<
            K, V, COMPARATOR >::wait (long long millisecs)
            throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.
InterruptedException if the wait is interrupted before it completes.
IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.647.3.25 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::wait () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlMap.h`

6.648 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 2671) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.

`#include <src/main/decaf/util/StlQueue.h>`Inheritance diagram for `decaf::util::StlQueue< T >`:

Data Structures

- class **QueueIterator**

Public Member Functions

- **StlQueue** ()
- virtual **~StlQueue** ()
- **Iterator**< T > * **iterator** ()
*Gets an **Iterator** (p. 1832) over this **Queue** (p. 2671).*
- void **clear** ()
Empties this queue.
- T & **front** ()
Returns a Reference to the element at the head of the queue.
- const T & **front** () const
Returns a Reference to the element at the head of the queue.
- T & **back** ()
Returns a Reference to the element at the tail of the queue.
- const T & **back** () const
Returns a Reference to the element at the tail of the queue.
- void **push** (const T &t)
Places a new Object at the Tail of the queue.
- void **enqueueFront** (const T &t)
Places a new Object at the front of the queue.
- T **pop** ()
Removes and returns the element that is at the Head of the queue.
- size_t **size** () const
*Gets the Number of elements currently in the **Queue** (p. 2671).*
- bool **empty** () const
*Checks if this **Queue** (p. 2671) is currently empty.*

- virtual std::vector< T > **toArray** () const
- void **reverse** (StlQueue< T > &target) const
Reverses the order of the contents of this queue and stores them in the target queue.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.
- T & **getSafeValue** ()
Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

6.648.1 Detailed Description

template<typename T> class decaf::util::StlQueue< T >

The **Queue** (p. 2671) class accepts messages with an psuh(m) command where m is the message to be queued. It destructively returns the message with **pop()** (p. 3047). **pop()** (p. 3047) returns messages in the order they were enqueued.

Queue (p.2671) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the pop method actually returns a reference to the element popped. This frees the app from having to call the **front** method before calling pop.

```
Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues a message
m string s = sq.pop(); // dequeues a message
```

= DESIGN CONSIDERATIONS

The **Queue** (p.2671) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p.2671).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

6.648.2 Constructor & Destructor Documentation

6.648.2.1 `template<typename T> decaf::util::StlQueue< T >::StlQueue ()`
[inline]

6.648.2.2 `template<typename T> virtual decaf::util::StlQueue< T >::~~StlQueue ()`
[inline, virtual]

6.648.3 Member Function Documentation

6.648.3.1 `template<typename T> const T& decaf::util::StlQueue< T >::back ()`
const [inline]

Returns a Reference to the element at the tail of the queue.

Returns:

reference to a queue type object or (safe)

6.648.3.2 `template<typename T> T& decaf::util::StlQueue< T >::back ()` [inline]

Returns a Reference to the element at the tail of the queue.

Returns:

reference to a queue type object or (safe)

6.648.3.3 `template<typename T> void decaf::util::StlQueue< T >::clear ()`
[inline]

Empties this queue.

6.648.3.4 `template<typename T> bool decaf::util::StlQueue< T >::empty () const`
[inline]

Checks if this **Queue** (p.2671) is currently empty.

Returns:

boolean indicating queue emptiness

6.648.3.5 `template<typename T> void decaf::util::StlQueue< T >::enqueueFront (const T & t) [inline]`

Places a new Object at the front of the queue.

Parameters:

t - **Queue** (p. 2671) Object Type reference.

6.648.3.6 `template<typename T> const T& decaf::util::StlQueue< T >::front () const [inline]`

Returns a Reference to the element at the head of the queue.

Returns:

reference to a queue type object or (safe)

6.648.3.7 `template<typename T> T& decaf::util::StlQueue< T >::front () [inline]`

Returns a Reference to the element at the head of the queue.

Returns:

reference to a queue type object or (safe)

6.648.3.8 `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue () [inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

Returns:

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back()`, `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front()`, and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop()`.

6.648.3.9 `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator () [inline]`

Gets an **Iterator** (p. 1832) over this **Queue** (p. 2671).

Returns:

new iterator pointer that is owned by the caller.

6.648.3.10 `template<typename T> virtual void decaf::util::StlQueue< T >::lock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.648.3.11 `template<typename T> virtual void decaf::util::StlQueue< T
>::notify () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

6.648.3.12 `template<typename T> virtual void decaf::util::StlQueue< T
>::notifyAll () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3125).

6.648.3.13 `template<typename T> T decaf::util::StlQueue< T >::pop () [inline]`

Removes and returns the element that is at the Head of the queue.

Returns:

reference to a queue type object or (safe)

6.648.3.14 `template<typename T> void decaf::util::StlQueue< T >::push (const T & t) [inline]`

Places a new Object at the Tail of the queue.

Parameters:

t - **Queue** (p. 2671) Object Type reference.

6.648.3.15 `template<typename T> void decaf::util::StlQueue< T >::reverse (StlQueue< T > & target) const [inline]`

Reverses the order of the contents of this queue and stores them in the target queue.

Parameters:

target - The target queue that will receive the contents of this queue in reverse order.

6.648.3.16 `template<typename T> size_t decaf::util::StlQueue< T >::size () const [inline]`

Gets the Number of elements currently in the **Queue** (p. 2671).

Returns:

Queue (p. 2671) Size

6.648.3.17 `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T >::toArray () const [inline, virtual]`

Returns:

the all values in this queue as a std::vector.

6.648.3.18 `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.648.3.19 `template<typename T> virtual void decaf::util::StlQueue< T >::unlock
() throw (decaf::lang::exceptions::RuntimeException) [inline,
virtual]`

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.648.3.20 `template<typename T> virtual void decaf::util::StlQueue<
T >::wait (long long millisecs, int nanos) throw
(decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.648.3.21 `template<typename T> virtual void decaf::util::StlQueue< T >::wait
(long long millisecs) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

```
6.648.3.22  template<typename T> virtual void decaf::util::StlQueue< T
>::wait () throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- src/main/decaf/util/StlQueue.h

6.649 decaf::util::StlSet< E > Class Template Reference

Set (p. 2905) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

`#include <src/main/decaf/util/StlSet.h>` Inheritance diagram for `decaf::util::StlSet< E >`:

Data Structures

- class **ConstSetIterator**
- class **SetIterator**

Public Member Functions

- **StlSet** ()
Default constructor - does nothing.
- **StlSet** (const **StlSet** &source)
Copy constructor - copies the content of the given set into this one.
- **StlSet** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual **~StlSet** ()
- **Iterator**< E > * **iterator** ()
Returns:
an iterator over a set of elements of type T.
- **Iterator**< E > * **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const
- virtual void **copy** (const **StlSet** &source)
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
*Removes all of the elements from this collection (optional operation).
The collection will be empty after this method returns.
This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1833) operation. Most implementations will probably choose to override this method for efficiency.
Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*
Exceptions:
***UnsupportedOperationException** if the clear operation is not supported by this collection*
- virtual bool **contains** (const E &value) const throw (lang::Exception)

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p. 1054).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

- virtual bool **isEmpty** () const
- virtual std::size_t **size** () const
- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1054) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.
IllegalArgumentException If the value is not a valid entry for this *Collection* (p. 1054).

6.649.1 Detailed Description

```
template<typename E> class decaf::util::StlSet< E >
```

Set (p. 2905) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

6.649.2 Constructor & Destructor Documentation

6.649.2.1 `template<typename E> decaf::util::StlSet< E >::StlSet () [inline]`

Default constructor - does nothing.

6.649.2.2 `template<typename E> decaf::util::StlSet< E >::StlSet (const StlSet< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.649.2.3 `template<typename E> decaf::util::StlSet< E >::StlSet (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.649.2.4 `template<typename E> virtual decaf::util::StlSet< E >::~~StlSet () [inline, virtual]`

6.649.3 Member Function Documentation

6.649.3.1 `template<typename E> virtual bool decaf::util::StlSet< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1054) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.1056).

```
6.649.3.2  template<typename E> virtual void decaf::util::StlSet< E >::clear ()
           throw ( lang::exceptions::UnsupportedOperationException ) [inline,
           virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.1833) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p.151).

6.649.3.3 `template<typename E> virtual bool decaf::util::StlSet< E >::contains
(const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p.1054).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p.152).

6.649.3.4 `template<typename E> virtual void decaf::util::StlSet< E >::copy (const
StlSet< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlSet< ActiveMQSession * >::StlSet()`.

6.649.3.5 `template<typename E> virtual bool decaf::util::StlSet< E >::equals
(const StlSet< E > & source) const [inline, virtual]`

6.649.3.6 `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty ()
const [inline, virtual]`

Returns:

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection< E >** (p.153).

6.649.3.7 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ()
const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p.1830).

6.649.3.8 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ()
[inline, virtual]`

Returns:

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p.1830).

```

6.649.3.9  template<typename E> virtual bool decaf::util::StlSet<
            E >::remove (const E & value) throw (
            lang::exceptions::UnsupportedOperationException,
            lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 1054).

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 155).

```

6.649.3.10  template<typename E> virtual std::size_t decaf::util::StlSet< E >::size
            () const [inline, virtual]

```

Returns:

The number of elements in this set.

Implements `decaf::util::Collection< E >` (p. 1062).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlSet.h`

6.650 activemq::wireformat::stomp::StompCommandConstants Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>
```

Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER_DESTINATION**
- static const std::string **HEADER_TRANSACTIONID**
- static const std::string **HEADER_CONTENTLENGTH**
- static const std::string **HEADER_SESSIONID**
- static const std::string **HEADER_RECEIPT_REQUIRED**
- static const std::string **HEADER_RECEIPTID**
- static const std::string **HEADER_MESSAGEID**
- static const std::string **HEADER_ACK**
- static const std::string **HEADER_LOGIN**
- static const std::string **HEADER_PASSWORD**
- static const std::string **HEADER_CLIENT_ID**
- static const std::string **HEADER_MESSAGE**
- static const std::string **HEADER_CORRELATIONID**
- static const std::string **HEADER_REQUESTID**
- static const std::string **HEADER_RESPONSEID**
- static const std::string **HEADER_EXPIRES**
- static const std::string **HEADER_PERSISTENT**
- static const std::string **HEADER_REPLYTO**
- static const std::string **HEADER_TYPE**
- static const std::string **HEADER_DISPATCH_ASYNC**
- static const std::string **HEADER_EXCLUSIVE**
- static const std::string **HEADER_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER_NOLOCAL**
- static const std::string **HEADER_PREFETCHSIZE**
- static const std::string **HEADER_JMSPRIORITY**
- static const std::string **HEADER_CONSUMERPRIORITY**
- static const std::string **HEADER_RETROACTIVE**
- static const std::string **HEADER_SUBSCRIPTIONNAME**
- static const std::string **HEADER_OLDSUBSCRIPTIONNAME**

- static const std::string **HEADER_TIMESTAMP**
- static const std::string **HEADER_REDELIVERED**
- static const std::string **HEADER_REDELIVERYCOUNT**
- static const std::string **HEADER_SELECTOR**
- static const std::string **HEADER_ID**
- static const std::string **HEADER_SUBSCRIPTION**
- static const std::string **HEADER_TRANSFORMATION**
- static const std::string **HEADER_TRANSFORMATION_ERROR**
- static const std::string **ACK_CLIENT**
- static const std::string **ACK_AUTO**
- static const std::string **ACK_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE_PREFIX**
- static const std::string **TOPIC_PREFIX**
- static const std::string **TEMPQUEUE_PREFIX**
- static const std::string **TEMPTOPIC_PREFIX**

6.650.1 Field Documentation

- 6.650.1.1** `const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT`
[static]
- 6.650.1.2** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK`
[static]
- 6.650.1.3** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_AUTO`
[static]
- 6.650.1.4** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT` [static]
- 6.650.1.5** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL` [static]
- 6.650.1.6** `const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN`
[static]
- 6.650.1.7** `const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES`
[static]
- 6.650.1.8** `const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT`
[static]
- 6.650.1.9** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT`
[static]
- 6.650.1.10** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED`
[static]
- 6.650.1.11** `const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT`
[static]
- 6.650.1.12** `const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD` [static]
- 6.650.1.13** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK` [static]
- 6.650.1.14** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID` [static]
-
- 6.650.1.15** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY` [static]

- `src/main/activemq/wireformat/stomp/StompCommandConstants.h`

6.651 activemq::wireformat::stomp::StompFrame Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

Public Member Functions

- **StompFrame** ()
Default constructor.
- virtual **~StompFrame** ()
Destruction.
- **StompFrame * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- void **copy** (const **StompFrame** *src)
Copies the contents of the passed Frame to this one.
- void **setCommand** (const std::string &cmd)
Sets the command for this stomp frame.
- const std::string & **getCommand** () const
Accessor for this frame's command field.
- bool **hasProperty** (const std::string &name) const
Checks if the given property is present in the Frame.
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const
Gets a property from this Frame's properties and returns it, or the default value given.
- std::string **removeProperty** (const std::string &name)
Gets and remove the property specified, if the property is not set, this method returns the empty string.
- void **setProperty** (const std::string &name, const std::string &value)
Sets the property given to the value specified in this Frame's Properties.
- **decaf::util::Properties** & **getProperties** ()
Gets access to the header properties for this frame.
- const **decaf::util::Properties** & **getProperties** () const
- const std::vector< unsigned char > & **getBody** () const
Accessor for the body data of this frame.
- std::vector< unsigned char > & **getBody** ()
Non-const version of the body accessor.

- `std::size_t getBodyLength () const`
Return the number of bytes contained in this frames body.
- `void setBody (const unsigned char *bytes, std::size_t numBytes)`
Sets the body data of this frame as a byte sequence.
- `void toStream (decaf::io::DataOutputStream *stream) const throw (decaf::io::IOException)`
Writes this Frame to an OuputStream in the Stomp Wire Format.
- `void fromStream (decaf::io::DataInputStream *stream) throw (decaf::io::IOException)`
Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

6.651.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

6.651.2 Constructor & Destructor Documentation

6.651.2.1 `activemq::wireformat::stomp::StompFrame::StompFrame () [inline]`

Default constructor.

6.651.2.2 `virtual activemq::wireformat::stomp::StompFrame::~~StompFrame () [inline, virtual]`

Destruction.

6.651.3 Member Function Documentation

6.651.3.1 `StompFrame* activemq::wireformat::stomp::StompFrame::clone () const`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

6.651.3.2 `void activemq::wireformat::stomp::StompFrame::copy (const StompFrame * src)`

Copies the contents of the passed Frame to this one.

Parameters:

src - Frame to copy

6.651.3.3 void activemq::wireformat::stomp::StompFrame::fromStream
(decaf::io::DataInputStream * *stream*) throw (decaf::io::IOException)

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

Parameters:

stream - The stream to read the Frame from.

Exceptions:

IOException if an error occurs while writing the Frame.

6.651.3.4 std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ()
[inline]

Non-const version of the body accessor.

6.651.3.5 const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () const
[inline]

Accessor for the body data of this frame.

Returns:

char pointer to body data

6.651.3.6 std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength ()
const [inline]

Return the number of bytes contained in this frames body.

Returns:

Body bytes length.

6.651.3.7 const std::string& activemq::wireformat::stomp::StompFrame::getCommand ()
const [inline]

Accessor for this frame's command field.

6.651.3.8 const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ()
const [inline]

6.651.3.9 decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ()
[inline]

Gets access to the header properties for this frame.

Returns:

the Properties object owned by this Frame

6.651.3.10 `std::string activemq::wireformat::stomp::StompFrame::getProperty
(const std::string & name, const std::string & fallback = "") const`
[inline]

Gets a property from this Frame's properties and returns it, or the default value given.

Parameters:

name - The name of the property to lookup

fallback - The default value to return if this value isn't set

Returns:

string value of the property asked for.

6.651.3.11 `bool activemq::wireformat::stomp::StompFrame::hasProperty (const
std::string & name) const` [inline]

Checks if the given property is present in the Frame.

Parameters:

name - The name of the property to check for.

6.651.3.12 `std::string activemq::wireformat::stomp::StompFrame::removeProperty
(const std::string & name)` [inline]

Gets and remove the property specified, if the property is not set, this method returns the empty string.

Parameters:

name - the Name of the property to get and return.

6.651.3.13 `void activemq::wireformat::stomp::StompFrame::setBody (const
unsigned char * bytes, std::size_t numBytes)`

Sets the body data of this frame as a byte sequence.

Parameters:

bytes The byte buffer to be set in the body.

numBytes The number of bytes in the buffer.

6.651.3.14 void activemq::wireformat::stomp::StompFrame::setCommand (const std::string & *cmd*) [inline]

Sets the command for this stomp frame.

Parameters:

cmd command The command to be set.

6.651.3.15 void activemq::wireformat::stomp::StompFrame::setProperty (const std::string & *name*, const std::string & *value*) [inline]

Sets the property given to the value specified in this Frame's Properties.

Parameters:

name - Name of the property.

value - Value to set the property to.

6.651.3.16 void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream * *stream*) const throw (decaf::io::IOException)

Writes this Frame to an OuputStream in the Stomp Wire Format.

Parameters:

stream - The stream to write the Frame to.

Exceptions:

IOException if an error occurs while reading the Frame.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompFrame.h**

6.652 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from StompFrame's.

```
#include <src/main/activemq/wireformat/stomp/StompHelper.h>
```

Public Member Functions

- **StompHelper** ()
- virtual **~StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)

Converts the Headers in a Stomp Frame into Headers in the given Message Command.
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)

Converts the Properties in a Message Command to Valid Headers and Properties in the StompFrame (p. 3061).
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)

Converts from a Stomp Destination to an ActiveMQDestination.
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

Converts from a ActiveMQDestination to a Stomp Destination Name.
- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)

Converts a MessageId instance to a Stomp MessageId String.
- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)

Converts a Stomp MessageId string to a MessageId.
- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)

Converts a ConsumerId instance to a Stomp ConsumerId String.
- **Pointer**< **ConsumerId** > **convertConsumerId** (const std::string &consumerId)

Converts a Stomp ConsumerId string to a ConsumerId.
- std::string **convertProducerId** (const **Pointer**< **ProducerId** > &producerId)

Converts a ProducerId instance to a Stomp ProducerId String.
- **Pointer**< **ProducerId** > **convertProducerId** (const std::string &producerId)

Converts a Stomp ProducerId string to a ProducerId.
- std::string **convertTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

Converts a TransactionId instance to a Stomp TransactionId String.
- **Pointer**< **TransactionId** > **convertTransactionId** (const std::string &transactionId)

Converts a Stomp TransactionId string to a TransactionId.

6.652.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since:

3.0

6.652.2 Constructor & Destructor Documentation

6.652.2.1 `activemq::wireformat::stomp::StompHelper::StompHelper ()` `[inline]`

6.652.2.2 `virtual activemq::wireformat::stomp::StompHelper::~~StompHelper ()`
`[inline, virtual]`

6.652.3 Member Function Documentation

6.652.3.1 `Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId (const std::string & consumerId)`

Converts a Stomp ConsumerId string to a ConsumerId.

Parameters:

consumerId - the String Consumer Id to convert.

Returns:

Pointer to a new ConsumerId.

6.652.3.2 `std::string activemq::wireformat::stomp::StompHelper::convertConsumerId (const Pointer< ConsumerId > & consumerId)`

Converts a ConsumerId instance to a Stomp ConsumerId String.

Parameters:

consumerId - the Consumer instance to convert.

Returns:

a Stomp Consumer Id String.

6.652.3.3 `std::string activemq::wireformat::stomp::StompHelper::convertDestination (const Pointer< ActiveMQDestination > & destination)`

Converts from a ActiveMQDestination to a Stomp Destination Name.

Parameters:

destination - The ActiveMQDestination to Convert

Returns:

the Stomp String name that defines the destination.

6.652.3.4 `Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination (const std::string & destination)`

Converts from a Stomp Destination to an ActiveMQDestination.

Parameters:

destination - The Stomp Destination name string.

Returns:

Pointer to a new ActiveMQDestination.

6.652.3.5 `Pointer<MessageId> activemq::wireformat::stomp::StompHelper::convertMessageId (const std::string & messageId)`

Converts a Stomp MessageId string to a MessageId.

Parameters:

messageId - the String message Id to convert.

Returns:

Pointer to a new MessageId.

6.652.3.6 `std::string activemq::wireformat::stomp::StompHelper::convertMessageId (const Pointer< MessageId > & messageId)`

Converts a MessageId instance to a Stomp MessageId String.

Parameters:

messageId - the MessageId instance to convert.

Returns:

a Stomp Message Id String.

6.652.3.7 `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId (const std::string & producerId)`

Converts a Stomp ProducerId string to a ProducerId.

Parameters:

producerId - the String Producer Id to convert.

Returns:

Pointer to a new ProducerId.

6.652.3.8 `std::string activemq::wireformat::stomp::StompHelper::convertProducerId (const Pointer< ProducerId > & producerId)`

Converts a ProducerId instance to a Stomp ProducerId String.

Parameters:

producerId - the Producer instance to convert.

Returns:

a Stomp Producer Id String.

6.652.3.9 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< Message > & message, const Pointer< StompFrame > & frame)`

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3061).

Parameters:

message - The message to move the Headers to.

frame - The frame to extract headers from.

6.652.3.10 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< StompFrame > & frame, const Pointer< Message > & message)`

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

Parameters:

frame - The frame to extract headers from.

message - The message to move the Headers to.

6.652.3.11 `Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId (const std::string & transactionId)`

Converts a Stomp TransactionId string to a TransactionId.

Parameters:

transactionId - the String Transaction Id to convert.

Returns:

Pointer to a new TransactionId.

6.652.3.12 `std::string activemq::wireformat::stomp::StompHelper::convertTransactionId (const Pointer< TransactionId > & transactionId)`

Converts a TransactionId instance to a Stomp TransactionId String.

Parameters:

transactionId - the Transaction instance to convert.

Returns:

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompHelper.h`

6.653 activemq::wireformat::stomp::StompWireFormat Class Reference

#include <src/main/activemq/wireformat/stomp/StompWireFormat.h> Inheritance diagram for activemq::wireformat::stomp::StompWireFormat:

Public Member Functions

- **StompWireFormat** ()
- virtual **~StompWireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out) throw (**decaf::io::IOException**)

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in) throw (**decaf::io::IOException**)

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

- virtual void **setVersion** (int version **AMQCPP_UNUSED**)

Set the Version.

- virtual int **getVersion** () const

Get the Version.

- virtual bool **inReceive** () const

Is there a Message being unmarshaled?

- virtual bool **hasNegotiator** () const

*Returns true if this **WireFormat** (p. 3363) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw (**decaf::lang::exceptions::UnsupportedOperationException**)

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

6.653.1 Constructor & Destructor Documentation

6.653.1.1 `activemq::wireformat::stomp::StompWireFormat::StompWireFormat ()`

6.653.1.2 `virtual
activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat ()
[virtual]`

6.653.2 Member Function Documentation

6.653.2.1 `virtual Pointer<transport::Transport> ac-
tivemq::wireformat::stomp::StompWireFormat::createNegotiator
(const Pointer< transport::Transport > & transport) throw (
decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns:

new instance of a **WireFormatNegotiator** (p. 3399).

Exceptions:

UnsupportedOperationException if the **WireFormat** (p. 3363) doesn't have a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3364).

6.653.2.2 `virtual int activemq::wireformat::stomp::StompWireFormat::getVersion
() const [inline, virtual]`

Get the Version.

Returns:

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3364).

6.653.2.3 `virtual bool ac-
tivemq::wireformat::stomp::StompWireFormat::hasNegotiator () const
[inline, virtual]`

Returns true if this **WireFormat** (p. 3363) has a Negotiator that needs to wrap the Transport that uses it.

Returns:

true if the **WireFormat** (p. 3363) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3364).

6.653.2.4 virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive
() const [inline, virtual]

Is there a Message being unmarshaled?

Returns:

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3365).

6.653.2.5 virtual void activemq::wireformat::stomp::StompWireFormat::marshal
(const Pointer< commands::Command > & *command*, const
activemq::transport::Transport * *transport*, decaf::io::DataOutputStream
* *out*) throw (decaf::io::IOException) [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters:

command The Command to Marshal to the output stream.

transport The Transport that initiated this marshal call.

out The output stream to write the command to.

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 3365).

6.653.2.6 virtual void activemq::wireformat::stomp::StompWireFormat::setVersion
(int version *AMQCPP_UNUSED*) [inline, virtual]

Set the Version.

Parameters:

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3365).

6.653.2.7 virtual Pointer<commands::Command> ac-
tivemq::wireformat::stomp::StompWireFormat::unmarshal
(const activemq::transport::Transport * *transport*,
decaf::io::DataInputStream * *in*) throw (decaf::io::IOException)
[virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form. Returns a Pointer to the newly unmarshaled Command.

Parameters:

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns:

the newly marshaled Command, caller owns the pointer

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 3366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormat.h`

6.654 activemq::wireformat::stomp::StompWireFormatFactory Class Reference

Factory used to create the Stomp Wire Format instance.

#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h> Inheritance diagram for activemq::wireformat::stomp::StompWireFormatFactory:

Public Member Functions

- **StompWireFormatFactory** ()
- virtual **~StompWireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw (**decaf::lang::exceptions::IllegalStateException**)
*Creates a new **WireFormat** (p. 3363) Object passing it a set of properties from which it can obtain any optional settings.*

6.654.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

6.654.2 Constructor & Destructor Documentation

6.654.2.1 **activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory** () [inline]

6.654.2.2 **virtual**
activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory () [inline, virtual]

6.654.3 Member Function Documentation

6.654.3.1 **virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) throw (**decaf::lang::exceptions::IllegalStateException**) [virtual]

Creates a new **WireFormat** (p. 3363) Object passing it a set of properties from which it can obtain any optional settings.

Parameters:

properties - the Properties for this **WireFormat** (p. 3363)

Implements **activemq::wireformat::WireFormatFactory** (p. 3367).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormatFactory.h**

6.655 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

#include <src/main/cms/Stoppable.h> Inheritance diagram for cms::Stoppable:

Public Member Functions

- virtual **~Stoppable** ()
- virtual void **stop** ()=0 throw (CMSEException)
Stops this service.

6.655.1 Detailed Description

Interface for a class that implements the stop method. An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

Since:

1.0

6.655.2 Constructor & Destructor Documentation

6.655.2.1 virtual cms::Stoppable::~~Stoppable () [inline, virtual]

6.655.3 Member Function Documentation

6.655.3.1 virtual void cms::Stoppable::stop () throw (CMSEException) [pure virtual]

Stops this service.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs while stopping the Service.

Implemented in **activemq::core::ActiveMQConnection** (p. 242).

The documentation for this class was generated from the following file:

- src/main/cms/**Stoppable.h**

6.656 decaf::util::logging::StreamHandler Class Reference

#include <src/main/decaf/util/logging/StreamHandler.h> Inheritance diagram for decaf::util::logging::StreamHandler:

Public Member Functions

- **StreamHandler** ()
*Create a **StreamHandler** (p. 3077), with no current output stream.*
- **StreamHandler** (io::OutputStream *stream, Formatter *formatter)
*Create a **StreamHandler** (p. 3077), with no current output stream.*
- virtual ~**StreamHandler** ()
- virtual void **close** () throw (decaf::io::IOException)
Close the current output stream.
- virtual void **flush** ()
Flush the Handler's output, clears any buffers.
- virtual void **publish** (const **LogRecord** &record)
*Publish the Log Record to this **Handler** (p. 1709).*
- virtual void **isLoggable** (const **LogRecord** &record)
*Check if this **Handler** (p. 1709) would actually log a given **LogRecord** (p. 2050).*
- virtual void **setFilter** (const **Filter** *filter)
*Sets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.*
- virtual const **Filter** * **getFilter** ()
*Gets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.*
- virtual void **setLevel** (**Level** level)
*Set (p. 2905) the log level specifying which message levels will be logged by this **Handler** (p. 1709).*
- virtual **Level** **getLevel** ()
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1709).*
- virtual void **setFormatter** (const **Formatter** *formatter)
*Sets the **Formatter** (p. 1699) used by this **Handler** (p. 1709).*
- virtual const **Formatter** * **getFormatter** ()
*Gets the **Formatter** (p. 1699) used by this **Handler** (p. 1709).*
- virtual io::OutputStream * **getOutputStream** () const
*Gets the output Stream that this **Handler** (p. 1709) is using.*

6.656.1 Constructor & Destructor Documentation

6.656.1.1 `decaf::util::logging::StreamHandler::StreamHandler ()` [inline]

Create a **StreamHandler** (p. 3077), with no current output stream.

6.656.1.2 `decaf::util::logging::StreamHandler::StreamHandler (io::OutputStream * stream, Formatter * formatter)` [inline]

Create a **StreamHandler** (p. 3077), with no current output stream.

References `decaf::util::logging::Fatal`.

6.656.1.3 `virtual decaf::util::logging::StreamHandler::~StreamHandler ()` [inline, virtual]

References `DECAF_CATCH_NOTHROW`, and `DECAF_CATCHALL_NOTHROW`.

6.656.2 Member Function Documentation

6.656.2.1 `virtual void decaf::util::logging::StreamHandler::close () throw (decaf::io::IOException)` [inline, virtual]

Close the current output stream. The close method will perform a flush and then close the **Handler** (p. 1709). After close has been called this **Handler** (p. 1709) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

Exceptions:

IOException

Implements `decaf::io::Closeable` (p. 1019).

6.656.2.2 `virtual void decaf::util::logging::StreamHandler::flush ()` [inline, virtual]

Flush the Handler's output, clears any buffers.

Implements `decaf::util::logging::Handler` (p. 1710).

6.656.2.3 `virtual const Filter* decaf::util::logging::StreamHandler::getFilter ()` [inline, virtual]

Gets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.

Returns:

`Filter` (p. 1630) derived instance

Implements `decaf::util::logging::Handler` (p. 1710).

6.656.2.4 `virtual const Formatter* decaf::util::logging::StreamHandler::getFormatter ()` [inline, virtual]

Gets the `Formatter` (p. 1699) used by this **Handler** (p. 1709).

Returns:

currently configured `Formatter` (p. 1699) derived instance

Implements `decaf::util::logging::Handler` (p. 1710).

6.656.2.5 `virtual Level decaf::util::logging::StreamHandler::getLevel ()` [inline, virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1709).

Returns:

Currently set `Level` enumeration value

Implements `decaf::util::logging::Handler` (p. 1710).

6.656.2.6 `virtual io::OutputStream* decaf::util::logging::StreamHandler::getOutputStream ()` const [inline, virtual]

Gets the output `Stream` that this **Handler** (p. 1709) is using.

Returns:

`OutputStream` pointer used by this handler.

6.656.2.7 `virtual void decaf::util::logging::StreamHandler::isLoggable (const LogRecord & record)` [inline, virtual]

Check if this **Handler** (p. 1709) would actually log a given **LogRecord** (p. 2050).

Parameters:

record `LogRecord` (p. 2050) to check

Implements `decaf::util::logging::Handler` (p. 1710).

6.656.2.8 `virtual void decaf::util::logging::StreamHandler::publish (const LogRecord & record)` [inline, virtual]

Publish the `Log Record` to this **Handler** (p. 1709).

Parameters:

record The `LogRecord` (p. 2050) to Publish

Implements `decaf::util::logging::Handler` (p. 1711).

References `DECAF_CATCH_RETHROW`, and `DECAF_CATCHALL_THROW`.

6.656.2.9 virtual void decaf::util::logging::StreamHandler::setFilter (const Filter * *filter*) [inline, virtual]

Sets the **Filter** (p. 1630) that this **Handler** (p. 1709) uses to filter Log Records.

Parameters:

filter Filter (p. 1630) derived instance

Implements **decaf::util::logging::Handler** (p. 1711).

6.656.2.10 virtual void decaf::util::logging::StreamHandler::setFormatter (const Formatter * *formatter*) [inline, virtual]

Sets the **Formatter** (p. 1699) used by this **Handler** (p. 1709).

Parameters:

formatter Formatter (p. 1699) derived instance

Implements **decaf::util::logging::Handler** (p. 1711).

6.656.2.11 virtual void decaf::util::logging::StreamHandler::setLevel (Level *level*) [inline, virtual]

Set (p. 2905) the log level specifying which message levels will be logged by this **Handler** (p. 1709). The intention is to allow developers to turn on verbose logging, but to limit the messages that are sent to certain Handlers.

Parameters:

level Level enumeration value

Implements **decaf::util::logging::Handler** (p. 1711).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

6.657 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 3081).

#include <src/main/cms/StreamMessage.h> Inheritance diagram for cms::StreamMessage:

Public Member Functions

- virtual **~StreamMessage** ()
- virtual bool **readBoolean** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the Stream message stream as a 1-byte value.
- virtual unsigned char **readByte** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Stream message stream.
- virtual void **writeByte** (unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the Stream message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the Stream message stream.
- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Char from the Stream message stream.

- virtual void **writeChar** (char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a char to the Stream message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit float from the Stream message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a float to the Stream message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the Stream message stream as a 8 byte value.

- virtual short **readShort** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Stream message stream.

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the Stream message stream as a 2 byte value.

- virtual int **readInt** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Stream message stream.

- virtual void **writeInt** (int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the Stream message stream as a 4 byte value.

- virtual long long **readLong** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Stream message stream.

- virtual void **writeLong** (long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the Stream message stream as a 8 byte value.

- virtual std::string **readString** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Stream message stream.

- virtual void **writeString** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Stream message stream.

6.657.1 Detailed Description

Interface for a **StreamMessage** (p.3081). The stream Messages provides a **Message** (p.2163) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

StreamMessage (p.3081) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p.1031). The string-to-primitive conversions may throw a runtime exception if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X		X
String	X	X	X		X	X	X	X	X	
byte[]										X

Since:

1.3

6.657.2 Constructor & Destructor Documentation

6.657.2.1 virtual cms::StreamMessage::~StreamMessage () [inline, virtual]

6.657.3 Member Function Documentation

6.657.3.1 virtual bool cms::StreamMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a Boolean from the Stream message stream.

Returns:

boolean value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 468).

6.657.3.2 virtual unsigned char cms::StreamMessage::readByte () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a Byte from the Stream message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 469).

6.657.3.3 `virtual std::size_t cms::StreamMessage::readBytes (unsigned char * buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a portion of the Stream message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 1031) is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 469).

6.657.3.4 `virtual std::size_t cms::StreamMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a byte array from the Stream message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 470).

6.657.3.5 `virtual char cms::StreamMessage::readChar () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a Char from the Stream message stream.

Returns:

char value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 470).

6.657.3.6 `virtual double cms::StreamMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a 64 bit double from the Stream message stream.

Returns:

double value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 471).

6.657.3.7 `virtual float cms::StreamMessage::readFloat () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a 32 bit float from the Stream message stream.

Returns:

double value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 471).

6.657.3.8 `virtual int cms::StreamMessage::readInt () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)` [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns:

int value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 472).

6.657.3.9 `virtual long long cms::StreamMessage::readLong () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit long from the Stream message stream.

Returns:

long long value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 472).

6.657.3.10 `virtual short cms::StreamMessage::readShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 16 bit signed short from the Stream message stream.

Returns:

short value from stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 472).

6.657.3.11 `virtual std::string cms::StreamMessage::readString () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads an ASCII String from the Stream message stream.

Returns:

String from stream

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 473).

6.657.3.12 `virtual unsigned short cms::StreamMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException) [pure virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2277) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2278) - if this type conversion is invalid.

MessageNotReadableException (p. 2330) - if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 473).

6.657.3.13 `virtual void cms::StreamMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a boolean to the Stream message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value boolean to write to the stream

Exceptions:

CMSEException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 474).

6.657.3.14 `virtual void cms::StreamMessage::writeByte (unsigned char value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[pure virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

Parameters:

value byte to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 474).

6.657.3.15 `virtual void cms::StreamMessage::writeBytes (const unsigned
char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Writes a portion of a byte array to the Stream message stream. size as the number of bytes to write.

Parameters:

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 475).

6.657.3.16 `virtual void cms::StreamMessage::writeBytes (const std::vector<
unsigned char > & value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters:

value bytes to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 475).

6.657.3.17 `virtual void cms::StreamMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a char to the Stream message stream as a 1-byte value.

Parameters:

value char to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 475).

6.657.3.18 `virtual void cms::StreamMessage::writeDouble (double value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a double to the Stream message stream as a 8 byte value.

Parameters:

value double to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 476).

6.657.3.19 `virtual void cms::StreamMessage::writeFloat (float value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a float to the Stream message stream as a 4 byte value.

Parameters:

value float to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 476).

6.657.3.20 `virtual void cms::StreamMessage::writeInt (int value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters:

value signed int to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 476).

6.657.3.21 `virtual void cms::StreamMessage::writeLong (long long value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a long long to the Stream message stream as a 8 byte value.

Parameters:

value signed long long to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 477).

6.657.3.22 `virtual void cms::StreamMessage::writeShort (short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters:

value signed short to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 477).

6.657.3.23 virtual void cms::StreamMessage::writeString (const std::string & *value*)
throw (cms::MessageNotWriteableException, cms::CMSException)
[pure virtual]

Writes an ASCII String to the Stream message stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 477).

6.657.3.24 virtual void cms::StreamMessage::writeUnsignedShort (unsigned short *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters:

value unsigned short to write to the stream

Exceptions:

CMSException (p. 1031) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 478).

The documentation for this class was generated from the following file:

- src/main/cms/StreamMessage.h

6.658 decaf::util::StringTokenizer Class Reference

```
#include <src/main/decaf/util/StringTokenizer.h>
```

Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)
Constructs a string tokenizer for the specified string.
- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const
Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.
- virtual bool **hasMoreTokens** () const
Tests if there are more tokens available from this tokenizer's string.
- virtual std::string **nextToken** () throw (lang::exceptions::NoSuchElementException)
Returns the next token from this string tokenizer.
- virtual std::string **nextToken** (const std::string &delim) throw (lang::exceptions::NoSuchElementException)
Returns the next token in this string tokenizer's string.
- virtual unsigned int **toArray** (std::vector< std::string > &array)
Grab all remaining tokens in the String and return them in the vector that is passed in by reference.
- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)
Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

6.658.1 Constructor & Destructor Documentation

6.658.1.1 decaf::util::StringTokenizer::StringTokenizer (const std::string & str, const std::string & delim = " \t\n\r\f", bool returnDelims = false)

Constructs a string tokenizer for the specified string. All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if delim is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p. 3094) may result in an Exception.

Parameters:

str - The string to tokenize

delim - String containing the delimiters

returnDelims - boolean indicating if the delimiters are returned as tokens

6.658.1.2 virtual decaf::util::StringTokenizer::~~StringTokenizer () [virtual]

6.658.2 Member Function Documentation

6.658.2.1 virtual int decaf::util::StringTokenizer::countTokens () const [virtual]

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception. The current position is not advanced.

Returns:

Count of remaining tokens

6.658.2.2 virtual bool decaf::util::StringTokenizer::hasMoreTokens () const [virtual]

Tests if there are more tokens available from this tokenizer's string.

Returns:

true if there are more tokens remaining

6.658.2.3 virtual std::string decaf::util::StringTokenizer::nextToken (const std::string & *delim*) throw (lang::exceptions::NoSuchElementException) [virtual]

Returns the next token in this string tokenizer's string. First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 3094) object is changed to be the characters in the string *delim*. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

Parameters:

delim - string containing the new set of delimiters

Returns:

next string in the token list

Exceptions:

NoSuchElementException

6.658.2.4 virtual std::string decaf::util::StringTokenizer::nextToken () throw (lang::exceptions::NoSuchElementException) [virtual]

Returns the next token from this string tokenizer.

Returns:

string value of next token

Exceptions:

NoSuchElementException

6.658.2.5 `virtual void decaf::util::StringTokenizer::reset (const std::string & str = "", const std::string & delim = "", bool returnDelims = false) [virtual]`

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning. This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenizer the string. If set to "", no change is made If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

Parameters:

str - New String to tokenize or "", defaults to ""

delim - New Delimiter String to use or "", defaults to ""

returnDelims - Should the Tokenizer return delimiters as Tokens, default false

6.658.2.6 `virtual unsigned int decaf::util::StringTokenizer::toArray (std::vector< std::string > & array) [virtual]`

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

Parameters:

array - vector to place token strings in

Returns:

number of string placed into the vector

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StringTokenizer.h**

6.659 activemq::commands::SubscriptionInfo Class Reference

#include <src/main/activemq/commands/SubscriptionInfo.h> Inheritance diagram for activemq::commands::SubscriptionInfo:

Public Member Functions

- **SubscriptionInfo** ()
- virtual **~SubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **SubscriptionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &subscribedDestination)

Static Public Attributes

- static const unsigned char **ID_SUBSCRIPTIONINFO** = 55

Protected Member Functions

- **SubscriptionInfo** (const **SubscriptionInfo** &)
- **SubscriptionInfo** & **operator=** (const **SubscriptionInfo** &)

Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subscriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

6.659.1 Constructor & Destructor Documentation

- 6.659.1.1** **activemq::commands::SubscriptionInfo::SubscriptionInfo** (const **SubscriptionInfo** &) [inline, protected]
- 6.659.1.2** **activemq::commands::SubscriptionInfo::SubscriptionInfo** ()
- 6.659.1.3** **virtual activemq::commands::SubscriptionInfo::~~SubscriptionInfo** () [virtual]

6.659.2 Member Function Documentation

- 6.659.2.1** **virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1461).

- 6.659.2.2** **virtual void activemq::commands::SubscriptionInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1462).

6.659.2.3 `virtual bool activemq::commands::SubscriptionInfo::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

6.659.2.4 `virtual std::string& activemq::commands::SubscriptionInfo::getClientId ()` [virtual]

6.659.2.5 `virtual const std::string& activemq::commands::SubscriptionInfo::getClientId () const` [virtual]

6.659.2.6 `virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.659.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ()`
[virtual]
- 6.659.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination () const`
[virtual]
- 6.659.2.9 `virtual std::string& activemq::commands::SubscriptionInfo::getSelector ()`
[virtual]
- 6.659.2.10 `virtual const std::string& activemq::commands::SubscriptionInfo::getSelector () const`
[virtual]
- 6.659.2.11 `virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ()`
[virtual]
- 6.659.2.12 `virtual const std::string& activemq::commands::SubscriptionInfo::getSubscriptionName () const`
[virtual]
- 6.659.2.13 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ()`
[virtual]
- 6.659.2.14 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () const` [virtual]
- 6.659.2.15 `SubscriptionInfo& activemq::commands::SubscriptionInfo::operator=(const SubscriptionInfo &)` [inline, protected]
- 6.659.2.16 `virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & clientId)` [virtual]
- 6.659.2.17 `virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.659.2.18 `virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & selector)` [virtual]
- 6.659.2.19 `virtual void activemq::commands::SubscriptionInfo::setSubscriptionName (const std::string & subscriptionName)` [virtual]
- 6.659.2.20 `virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & subscribedDestination)` [virtual]
- 6.659.2.21 `virtual std::string activemq::commands::SubscriptionInfo::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 718).

6.659.3 Field Documentation

6.659.3.1 `std::string activemq::commands::SubscriptionInfo::clientId` [protected]

6.659.3.2 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination`
[protected]

6.659.3.3 `const unsigned char activemq::commands::SubscriptionInfo::ID__ - SUBSCRIPTIONINFO = 55` [static]

6.659.3.4 `std::string activemq::commands::SubscriptionInfo::selector` [protected]

6.659.3.5 `std::string activemq::commands::SubscriptionInfo::subscriptionName`
[protected]

6.659.3.6 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

6.660 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3102).

#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.660.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3102). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.660.2 Constructor & Destructor Documentation

6.660.2.1 `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

6.660.2.2 `virtual activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

6.660.3 Member Function Documentation

6.660.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.660.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.660.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.660.3.4 virtual void **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.660.3.5 virtual int **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.660.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.660.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SubscriptionInfoMarshaller.h**

6.661 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3106).

#include <src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.661.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3106). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.661.2 Constructor & Destructor Documentation

6.661.2.1 `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

6.661.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

6.661.3 Member Function Documentation

6.661.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.661.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.661.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.661.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.661.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.661.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.661.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SubscriptionInfoMarshaller.h**

6.662 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3110).

#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.662.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3110). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.662.2 Constructor & Destructor Documentation

6.662.2.1 `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

6.662.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

6.662.3 Member Function Documentation

6.662.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.662.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.662.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.662.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.662.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.662.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.662.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h

6.663 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3114).

#include <src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.663.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3114). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.663.2 Constructor & Destructor Documentation

6.663.2.1 `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

6.663.2.2 `virtual activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

6.663.3 Member Function Documentation

6.663.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.663.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.663.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.663.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.663.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.663.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.663.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h

6.664 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3118).

#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.664.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3118). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.664.2 Constructor & Destructor Documentation

6.664.2.1 `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` `[inline]`

6.664.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` `[inline, virtual]`

6.664.3 Member Function Documentation

6.664.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.664.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.664.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.664.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.664.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.664.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1448).

6.664.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

6.665 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

#include <src/main/decaf/util/concurrent/Synchronizable.h> Inheritance diagram for decaf::util::concurrent::Synchronizable:

Public Member Functions

- virtual **~Synchronizable** ()
- virtual void **lock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p. 2023) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.665.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since:

1.0

6.665.2 Constructor & Destructor Documentation

6.665.2.1 virtual decaf::util::concurrent::Synchronizable::~~Synchronizable ()
[inline, virtual]

6.665.3 Member Function Documentation

6.665.3.1 virtual void decaf::util::concurrent::Synchronizable::lock () throw (
decaf::lang::exceptions::RuntimeException) [pure virtual]

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2227),
`decaf::internal::io::StandardErrorOutputStream` (p. 2996), `de-`
`caf::internal::io::StandardInputStream` (p. 3002), `decaf::internal::io::StandardOutputStream`
(p. 3009), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3134), `de-`
`caf::io::BlockingByteArrayInputStream` (p. 726), `decaf::io::ByteArrayInputStream`
(p. 895), `decaf::io::ByteArrayOutputStream` (p. 902), `decaf::io::FilterInputStream`
(p. 1634), `decaf::io::FilterOutputStream` (p. 1643), `decaf::net::SocketInputStream`
(p. 2979), `decaf::net::SocketOutputStream` (p. 2986), `decaf::util::AbstractCollection<`
`E >` (p. 154), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARA-`
`TOR >` (p. 1110), `decaf::util::concurrent::Mutex` (p. 2386), `decaf::util::StlMap<`
`K, V, COMPARATOR >` (p. 3037), `decaf::util::StlQueue< T >`
(p. 3047), `decaf::util::AbstractCollection< transport::TransportListener *`
`>` (p. 154), `decaf::util::AbstractCollection< Pointer< Synchronization >`
`>` (p. 154), `decaf::util::AbstractCollection< CompositeTask * >` (p. 154),
`decaf::util::AbstractCollection< URI >` (p. 154), `decaf::util::AbstractCollection<`
`ActiveMQSession * >` (p. 154), `decaf::util::AbstractCollection< Pointer<`
`DestinationInfo > >` (p. 154), `decaf::util::AbstractCollection< Primitive-`
`ValueNode >` (p. 154), `decaf::util::AbstractCollection< Pointer< Com-`
`mand > >` (p. 154), `decaf::util::AbstractCollection< Pointer< Back-`
`upTransport > >` (p. 154), `decaf::util::concurrent::ConcurrentStlMap<`
`Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR`
`>` (p. 1110), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-`
`tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR`
`>` (p. 1110), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-`
`sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR`
`>` (p. 1110), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId`
`>, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1110),
`decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,`
`Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1110),
`decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<`

ProducerState >, ProducerId::COMPARATOR > (p.1110), decaf::util::StlMap< cms::Session *, SessionResolver * > (p.3037), decaf::util::StlMap< std::string, WireFormatFactory * > (p.3037), decaf::util::StlMap< std::string, PrimitiveValueNode > (p.3037), decaf::util::StlMap< std::string, cms::Queue * > (p.3037), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p.3037), decaf::util::StlMap< std::string, CachedConsumer * > (p.3037), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p.3037), decaf::util::StlMap< std::string, TransportFactory * > (p.3037), decaf::util::StlMap< int, Pointer< Command > > (p.3037), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p.3037), decaf::util::StlMap< std::string, CachedProducer * > (p.3037), decaf::util::StlMap< std::string, cms::Topic * > (p.3037), decaf::util::StlQueue< Pointer< Transport > > (p.3047), decaf::util::StlQueue< Pointer< MessageDispatch > > (p.3047), decaf::util::StlQueue< Task > (p.3047), decaf::util::StlQueue< Pointer< Command > > (p.3047), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p.3047).

Referenced by decaf::util::concurrent::Lock::lock().

6.665.3.2 virtual void decaf::util::concurrent::Synchronizable::notify
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p.3122) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p.2227),
decaf::internal::io::StandardErrorOutputStream (p.2996), **decaf::internal::io::StandardInputStream** (p.3003), **decaf::internal::io::StandardOutputStream** (p.3010), **decaf::internal::util::concurrent::SynchronizableImpl** (p.3134), **decaf::io::BlockingByteArrayInputStream** (p.727), **decaf::io::ByteArrayInputStream** (p.895), **decaf::io::ByteArrayOutputStream** (p.902), **decaf::io::FilterInputStream** (p.1635), **decaf::io::FilterOutputStream** (p.1643), **decaf::net::SocketInputStream** (p.2980), **decaf::net::SocketOutputStream** (p.2986), **decaf::util::AbstractCollection< E >** (p.154), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p.1110), **decaf::util::concurrent::Mutex** (p.2386), **decaf::util::StlMap< K, V, COMPARATOR >** (p.3038), **decaf::util::StlQueue< T >** (p.3047), **decaf::util::AbstractCollection< transport::TransportListener * >** (p.154), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p.154), **decaf::util::AbstractCollection< CompositeTask * >** (p.154), **decaf::util::AbstractCollection< URI >** (p.154), **decaf::util::AbstractCollection< ActiveMQSession * >** (p.154), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p.154), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p.154), **decaf::util::AbstractCollection< Pointer< Command > >** (p.154), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p.154), **decaf::util::concurrent::ConcurrentStlMap<**

Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR
 > (p. 1110), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
 > (p. 1110), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
 > (p. 1110), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1110),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,
 Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1110),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
 ProducerState >, ProducerId::COMPARATOR > (p. 1110), decaf::util::StlMap<
 cms::Session *, SessionResolver * > (p. 3038), decaf::util::StlMap< std::string,
 WireFormatFactory * > (p. 3038), decaf::util::StlMap< std::string, PrimitiveVal-
 ueNode > (p. 3038), decaf::util::StlMap< std::string, cms::Queue * > (p. 3038),
 decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-
 mands::ProducerId::COMPARATOR > (p. 3038), decaf::util::StlMap< std::string,
 CachedConsumer * > (p. 3038), decaf::util::StlMap< Pointer< commands::ConsumerId
 >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3038),
 decaf::util::StlMap< std::string, TransportFactory * > (p. 3038), decaf::util::StlMap<
 int, Pointer< Command > > (p. 3038), decaf::util::StlMap< Pointer< com-
 mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR
 > (p. 3038), decaf::util::StlMap< std::string, CachedProducer * > (p. 3038),
 decaf::util::StlMap< std::string, cms::Topic * > (p. 3038), decaf::util::StlQueue<
 Pointer< Transport > > (p. 3047), decaf::util::StlQueue< Pointer< MessageDis-
 patch > > (p. 3047), decaf::util::StlQueue< Task > (p. 3047), decaf::util::StlQueue<
 Pointer< Command > > (p. 3047), and decaf::util::StlQueue< decaf::lang::Pointer<
 commands::MessageDispatch > > (p. 3047).

6.665.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the **Syn-
chronizable** (p. 3122) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2227),
decaf::internal::io::StandardOutputStream (p. 2996), **de-
 caf::internal::io::StandardInputStream** (p. 3003), **decaf::internal::io::StandardOutputStream**
 (p. 3010), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3134), **de-
 caf::io::BlockingByteArrayInputStream** (p. 727), **decaf::io::ByteArrayInputStream**
 (p. 896), **decaf::io::ByteArrayOutputStream** (p. 903), **decaf::io::FilterInputStream**
 (p. 1635), **decaf::io::FilterOutputStream** (p. 1643), **decaf::net::SocketInputStream**
 (p. 2980), **decaf::net::SocketOutputStream** (p. 2986), **decaf::util::AbstractCollection<
 E >** (p. 154), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARA-
 TOR >** (p. 1110), **decaf::util::concurrent::Mutex** (p. 2387), **decaf::util::StlMap<
 K, V, COMPARATOR >** (p. 3038), **decaf::util::StlQueue< T >**
 (p. 3047), **decaf::util::AbstractCollection< transport::TransportListener ***

> (p. 154), `decaf::util::AbstractCollection< Pointer< Synchronization >`
> (p. 154), `decaf::util::AbstractCollection< CompositeTask * >` (p. 154),
`decaf::util::AbstractCollection< URI >` (p. 154), `decaf::util::AbstractCollection<`
`ActiveMQSession * >` (p. 154), `decaf::util::AbstractCollection< Pointer<`
`DestinationInfo > >` (p. 154), `decaf::util::AbstractCollection< Primitive-`
`ValueNode >` (p. 154), `decaf::util::AbstractCollection< Pointer< Com-`
`mand > >` (p. 154), `decaf::util::AbstractCollection< Pointer< Back-`
`upTransport > >` (p. 154), `decaf::util::concurrent::ConcurrentStlMap<`
`Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR`
> (p. 1110), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-`
`tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR`
> (p. 1110), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-`
`sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR`
> (p. 1110), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId`
>, `Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1110),
`decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,`
`Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1110),
`decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<`
`ProducerState >, ProducerId::COMPARATOR >` (p. 1110), `decaf::util::StlMap<`
`cms::Session *, SessionResolver * >` (p. 3038), `decaf::util::StlMap< std::string,`
`WireFormatFactory * >` (p. 3038), `decaf::util::StlMap< std::string, PrimitiveVal-`
`ueNode >` (p. 3038), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3038),
`decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-`
`mands::ProducerId::COMPARATOR >` (p. 3038), `decaf::util::StlMap< std::string,`
`CachedConsumer * >` (p. 3038), `decaf::util::StlMap< Pointer< commands::ConsumerId`
>, `ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3038),
`decaf::util::StlMap< std::string, TransportFactory * >` (p. 3038), `decaf::util::StlMap<`
`int, Pointer< Command > >` (p. 3038), `decaf::util::StlMap< Pointer< com-`
`mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR`
> (p. 3038), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3038),
`decaf::util::StlMap< std::string, cms::Topic * >` (p. 3038), `decaf::util::StlQueue<`
`Pointer< Transport > >` (p. 3047), `decaf::util::StlQueue< Pointer< MessageDis-`
`patch > >` (p. 3047), `decaf::util::StlQueue< Task >` (p. 3047), `decaf::util::StlQueue<`
`Pointer< Command > >` (p. 3047), and `decaf::util::StlQueue< decaf::lang::Pointer<`
`commands::MessageDispatch > >` (p. 3047).

6.665.3.4 `virtual bool decaf::util::concurrent::Synchronizable::tryLock () throw (`
`decaf::lang::exceptions::RuntimeException) [pure virtual]`

Attempts to **Lock** (p. 2023) the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2228),
`decaf::internal::io::StandardOutputStream` (p. 2996), `de-`
`cafe::internal::io::StandardInputStream` (p. 3005), `decaf::internal::io::StandardOutputStream`
(p. 3010), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3135), `de-`
`cafe::io::BlockingByteArrayInputStream` (p. 730), `decaf::io::ByteArrayInputStream`

(p. 898), decaf::io::ByteArrayOutputStream (p. 904), decaf::io::FilterInputStream (p. 1637), decaf::io::FilterOutputStream (p. 1643), decaf::net::SocketInputStream (p. 2982), decaf::net::SocketOutputStream (p. 2987), decaf::util::AbstractCollection< E > (p. 157), decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1115), decaf::util::concurrent::Mutex (p. 2387), decaf::util::StlMap< K, V, COMPARATOR > (p. 3040), decaf::util::StlQueue< T > (p. 3048), decaf::util::AbstractCollection< transport::TransportListener * > (p. 157), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 157), decaf::util::AbstractCollection< CompositeTask * > (p. 157), decaf::util::AbstractCollection< URI > (p. 157), decaf::util::AbstractCollection< ActiveMQSession * > (p. 157), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 157), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 157), decaf::util::AbstractCollection< Pointer< Command > > (p. 157), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 157), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1115), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3040), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3040), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3040), decaf::util::StlMap< std::string, cms::Queue * > (p. 3040), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, TransportFactory * > (p. 3040), decaf::util::StlMap< int, Pointer< Command > > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedProducer * > (p. 3040), decaf::util::StlMap< std::string, cms::Topic * > (p. 3040), decaf::util::StlQueue< Pointer< Transport > > (p. 3048), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3048), decaf::util::StlQueue< Task > (p. 3048), decaf::util::StlQueue< Pointer< Command > > (p. 3048), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3048).

6.665.3.5 virtual void decaf::util::concurrent::Synchronizable::unlock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2229),
`decaf::internal::io::StandardErrorOutputStream` (p. 2997), `de-`

caf::internal::io::StandardInputStream (p. 3005), decaf::internal::io::StandardOutputStream (p. 3010), decaf::internal::util::concurrent::SynchronizableImpl (p. 3135), decaf::io::BlockingByteArrayInputStream (p. 730), decaf::io::ByteArrayInputStream (p. 898), decaf::io::ByteArrayOutputStream (p. 904), decaf::io::FilterInputStream (p. 1637), decaf::io::FilterOutputStream (p. 1644), decaf::net::SocketInputStream (p. 2982), decaf::net::SocketOutputStream (p. 2987), decaf::util::AbstractCollection< E > (p. 157), decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1115), decaf::util::concurrent::Mutex (p. 2387), decaf::util::StlMap< K, V, COMPARATOR > (p. 3040), decaf::util::StlQueue< T > (p. 3049), decaf::util::AbstractCollection< transport::TransportListener * > (p. 157), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 157), decaf::util::AbstractCollection< CompositeTask * > (p. 157), decaf::util::AbstractCollection< URI > (p. 157), decaf::util::AbstractCollection< ActiveMQSession * > (p. 157), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 157), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 157), decaf::util::AbstractCollection< Pointer< Command > > (p. 157), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 157), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1115), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1115), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3040), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3040), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3040), decaf::util::StlMap< std::string, cms::Queue * > (p. 3040), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, TransportFactory * > (p. 3040), decaf::util::StlMap< int, Pointer< Command > > (p. 3040), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 3040), decaf::util::StlMap< std::string, CachedProducer * > (p. 3040), decaf::util::StlMap< std::string, cms::Topic * > (p. 3040), decaf::util::StlQueue< Pointer< Transport > > (p. 3049), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3049), decaf::util::StlQueue< Task > (p. 3049), decaf::util::StlQueue< Pointer< Command > > (p. 3049), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3049).

Referenced by decaf::util::concurrent::Lock::unlock(), and decaf::util::concurrent::Lock::~~Lock().

6.665.3.6 virtual void decaf::util::concurrent::Synchronizable::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2229), **decaf::internal::io::StandardOutputStream** (p. 2997), **decaf::internal::io::StandardInputStream** (p. 3005), **decaf::internal::io::StandardOutputStream** (p. 3011), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 3135), **decaf::io::BlockingByteArrayInputStream** (p. 730), **decaf::io::ByteArrayInputStream** (p. 898), **decaf::io::ByteArrayOutputStream** (p. 905), **decaf::io::FilterInputStream** (p. 1638), **decaf::io::FilterOutputStream** (p. 1644), **decaf::net::SocketInputStream** (p. 2982), **decaf::net::SocketOutputStream** (p. 2987), **decaf::util::AbstractCollection< E >** (p. 158), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1115), **decaf::util::concurrent::Mutex** (p. 2388), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3041), **decaf::util::StlQueue< T >** (p. 3049), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 158), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 158), **decaf::util::AbstractCollection< CompositeTask * >** (p. 158), **decaf::util::AbstractCollection< URI >** (p. 158), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 158), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 158), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 158), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 158), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 158), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1115), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1115), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR**

> (p. 1115), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1115), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1115), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1115), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3041), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3041), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3041), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3041), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3041), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3041), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3041), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3041), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3041), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3041), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3041), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3041), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3049), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3049), `decaf::util::StlQueue< Task >` (p. 3049), `decaf::util::StlQueue< Pointer< Command > >` (p. 3049), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3049).

6.665.3.7 `virtual void decaf::util::concurrent::Synchronizable::wait (long long milliseconds) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)` [pure virtual]

Waits on a signal from this object, which is generated by a call to `Notify`. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

milliseconds the time in milliseconds to wait, or `WAIT_INFINITE`

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2229), `decaf::internal::io::StandardOutputStream` (p. 2998), `decaf::internal::io::StandardInputStream` (p. 3006), `decaf::internal::io::StandardOutputStream` (p. 3011), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3136), `decaf::io::BlockingByteArrayInputStream` (p. 731), `decaf::io::ByteArrayInputStream` (p. 899), `decaf::io::ByteArrayOutputStream` (p. 905), `decaf::io::FilterInputStream` (p. 1638), `decaf::io::FilterOutputStream` (p. 1644), `decaf::net::SocketInputStream` (p. 2983), `decaf::net::SocketOutputStream` (p. 2988), `decaf::util::AbstractCollection< E >` (p. 158), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1116), `decaf::util::concurrent::Mutex` (p. 2388), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3041), `decaf::util::StlQueue< T >`

(p. 3049), decaf::util::AbstractCollection< transport::TransportListener * > (p. 158), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 158), decaf::util::AbstractCollection< CompositeTask * > (p. 158), decaf::util::AbstractCollection< URI > (p. 158), decaf::util::AbstractCollection< ActiveMQSession * > (p. 158), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 158), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 158), decaf::util::AbstractCollection< Pointer< Command > > (p. 158), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 158), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > > (p. 1116), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > > (p. 1116), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > > (p. 1116), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > > (p. 1116), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > > (p. 1116), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > > (p. 1116), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 3041), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 3041), decaf::util::StlMap< std::string, PrimitiveValueNode > > (p. 3041), decaf::util::StlMap< std::string, cms::Queue * > (p. 3041), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > > (p. 3041), decaf::util::StlMap< std::string, CachedConsumer * > (p. 3041), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > > (p. 3041), decaf::util::StlMap< std::string, TransportFactory * > (p. 3041), decaf::util::StlMap< int, Pointer< Command > > (p. 3041), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > > (p. 3041), decaf::util::StlMap< std::string, CachedProducer * > > (p. 3041), decaf::util::StlMap< std::string, cms::Topic * > > (p. 3041), decaf::util::StlQueue< Pointer< Transport > > > (p. 3049), decaf::util::StlQueue< Pointer< MessageDispatch > > > (p. 3049), decaf::util::StlQueue< Task > > (p. 3049), decaf::util::StlQueue< Pointer< Command > > > (p. 3049), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > > (p. 3049).

6.665.3.8 virtual void decaf::util::concurrent::Synchronizable::wait
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 3122) Object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2230),
`decaf::internal::io::StandardErrorOutputStream` (p. 2998), `de-`
`cafe::internal::io::StandardInputStream` (p. 3006), `decaf::internal::io::StandardOutputStream`
(p. 3012), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 3136), `de-`
`cafe::io::BlockingByteArrayInputStream` (p. 731), `decaf::io::ByteArrayInputStream`
(p. 899), `decaf::io::ByteArrayOutputStream` (p. 906), `decaf::io::FilterInputStream`
(p. 1639), `decaf::io::FilterOutputStream` (p. 1645), `decaf::net::SocketInputStream`
(p. 2983), `decaf::net::SocketOutputStream` (p. 2988), `decaf::util::AbstractCollection<`
`E >` (p. 159), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARA-`
`TOR >` (p. 1116), `decaf::util::concurrent::Mutex` (p. 2389), `decaf::util::StlMap<`
`K, V, COMPARATOR >` (p. 3042), `decaf::util::StlQueue< T >`
(p. 3050), `decaf::util::AbstractCollection< transport::TransportListener *`
`>` (p. 159), `decaf::util::AbstractCollection< Pointer< Synchronization >`
`>` (p. 159), `decaf::util::AbstractCollection< CompositeTask * >` (p. 159),
`decaf::util::AbstractCollection< URI >` (p. 159), `decaf::util::AbstractCollection<`
`ActiveMQSession * >` (p. 159), `decaf::util::AbstractCollection< Pointer<`
`DestinationInfo > >` (p. 159), `decaf::util::AbstractCollection< Primitive-`
`ValueNode >` (p. 159), `decaf::util::AbstractCollection< Pointer< Com-`
`mand > >` (p. 159), `decaf::util::AbstractCollection< Pointer< Back-`
`upTransport > >` (p. 159), `decaf::util::concurrent::ConcurrentStlMap<`
`Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR`
`>` (p. 1116), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-`
`tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR`
`>` (p. 1116), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-`
`sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR`
`>` (p. 1116), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId`
`>, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1116),
`decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >,`
`Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1116),
`decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<`
`ProducerState >, ProducerId::COMPARATOR >` (p. 1116), `decaf::util::StlMap<`
`cms::Session *, SessionResolver * >` (p. 3042), `decaf::util::StlMap< std::string,`
`WireFormatFactory * >` (p. 3042), `decaf::util::StlMap< std::string, PrimitiveVal-`
`ueNode >` (p. 3042), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3042),
`decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-`
`mands::ProducerId::COMPARATOR >` (p. 3042), `decaf::util::StlMap< std::string,`
`CachedConsumer * >` (p. 3042), `decaf::util::StlMap< Pointer< commands::ConsumerId`
`>, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3042),
`decaf::util::StlMap< std::string, TransportFactory * >` (p. 3042), `decaf::util::StlMap<`
`int, Pointer< Command > >` (p. 3042), `decaf::util::StlMap< Pointer< com-`
`mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR`
`>` (p. 3042), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3042),
`decaf::util::StlMap< std::string, cms::Topic * >` (p. 3042), `decaf::util::StlQueue<`
`Pointer< Transport > >` (p. 3050), `decaf::util::StlQueue< Pointer< MessageDis-`
`patch > >` (p. 3050), `decaf::util::StlQueue< Task >` (p. 3050), `decaf::util::StlQueue<`
`Pointer< Command > >` (p. 3050), and `decaf::util::StlQueue< decaf::lang::Pointer<`
`commands::MessageDispatch > >` (p. 3050).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Synchronizable.h`

6.666 decaf::internal::util::concurrent::SynchronizableImpl Class Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

#include <src/main/decaf/internal/util/concurrent/SynchronizableImpl.h> Inheritance diagram for decaf::internal::util::concurrent::SynchronizableImpl:

Public Member Functions

- **SynchronizableImpl** ()
- virtual **~SynchronizableImpl** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.666.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Since:

1.0

6.666.2 Constructor & Destructor Documentation

6.666.2.1 `decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl()`

6.666.2.2 `virtual decaf::internal::util::concurrent::SynchronizableImpl::~SynchronizableImpl() [virtual]`

6.666.3 Member Function Documentation

6.666.3.1 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Locks the object.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3123).

6.666.3.2 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 3124).

6.666.3.3 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3125).

6.666.3.4 virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock() () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns:

true if the lock was acquired, false if it is already held by another thread.

Exceptions:

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3126).

6.666.3.5 virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock() () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Unlocks the object.

Exceptions:

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3127).

6.666.3.6 virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions:

IllegalArgumentException if an error occurs or the *nanos* argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3128).

6.666.3.7 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3130).

6.666.3.8 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3131).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/SynchronizableImpl.h`

6.667 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 3137), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

Public Member Functions

- virtual **~Synchronization** ()
- virtual void **beforeEnd** ()=0 throw (exceptions::ActiveMQException)
- virtual void **afterCommit** ()=0 throw (exceptions::ActiveMQException)
- virtual void **afterRollback** ()=0 throw (exceptions::ActiveMQException)

6.667.1 Detailed Description

Transacted Object **Synchronization** (p. 3137), used to sync the events of a Transaction with the items in the Transaction.

6.667.2 Constructor & Destructor Documentation

6.667.2.1 virtual activemq::core::Synchronization::~~Synchronization () [inline, virtual]

6.667.3 Member Function Documentation

6.667.3.1 virtual void activemq::core::Synchronization::afterCommit () throw (exceptions::ActiveMQException) [pure virtual]

6.667.3.2 virtual void activemq::core::Synchronization::afterRollback () throw (exceptions::ActiveMQException) [pure virtual]

6.667.3.3 virtual void activemq::core::Synchronization::beforeEnd () throw (exceptions::ActiveMQException) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Synchronization.h**

6.668 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A `BlockingQueue` blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

```
#include <src/main/decaf/util/concurrent/SynchronousQueue.h>
```

Data Structures

- class `EmptyIterator`

Public Member Functions

- `SynchronousQueue` ()
- virtual `~SynchronousQueue` ()
- virtual void **put** (const E &value) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Adds the specified element to this queue, waiting if necessary for another thread to receive it.
- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.
- virtual bool **offer** (const E &value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into this queue, if another thread is waiting to receive it.
- virtual E **take** () throw (decaf::lang::exceptions::InterruptedException)
Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException)
Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.
- virtual bool **poll** (E &result)
Retrieves and removes the head of this queue, if another thread is currently making an element available.
- virtual bool **equals** (const **Collection**< E > &value) const
- virtual `decaf::util::Iterator`< E > * **iterator** ()
- virtual `decaf::util::Iterator`< E > * **iterator** () const
- virtual bool **isEmpty** () const
- virtual std::size_t **size** () const
- virtual int **remainingCapacity** () const

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
- virtual bool **contains** (const E &value DECAF_UNUSED) const throw (lang::Exception)
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw (lang::Exception)
- virtual bool **remove** (const E &value DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **removeAll** (const **Collection**< E > &collection DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **retainAll** (const **Collection**< E > &collection DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **peek** (E &result DECAF_UNUSED) const
- virtual std::vector< E > **toArray** () const
- virtual std::size_t **drainTo** (**Collection**< E > &c) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)
- virtual std::size_t **drainTo** (**Collection**< E > &c, std::size_t maxElements) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

6.668.1 Detailed Description

template<typename E> class decaf::util::concurrent::SynchronousQueue< E >

A `BlockingQueue` `blocking queue` in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa. A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot **peek** at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and `poll()` (p.3142) will return `null`. For purposes of other **Collection** (p.1054) methods (for example **contains**), a **SynchronousQueue** (p.3138) acts as an empty collection. This queue does not permit `null` elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to `true` grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the **Collection** (p.1054) and **Iterator** (p.1832) interfaces.

Since:

1.0

6.668.2 Constructor & Destructor Documentation

6.668.2.1 `template<typename E > decaf::util::concurrent::SynchronousQueue< E >::SynchronousQueue () [inline]`

6.668.2.2 `template<typename E > virtual decaf::util::concurrent::SynchronousQueue< E >::~~SynchronousQueue () [inline, virtual]`

6.668.3 Member Function Documentation

6.668.3.1 `template<typename E > virtual void decaf::util::concurrent::SynchronousQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

6.668.3.2 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::contains (const E &value DECAF_UNUSED) const throw (lang::Exception) [inline, virtual]`

6.668.3.3 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception) [inline, virtual]`

6.668.3.4 `template<typename E > virtual std::size_t decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E > & c, std::size_t maxElements) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

References `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

6.668.3.5 `template<typename E > virtual std::size_t decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E > & c) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

References `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

- 6.668.3.6** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::equals
(const Collection< E > & value) const [inline, virtual]`
- 6.668.3.7** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::isEmpty
() const [inline, virtual]`
- 6.668.3.8** `template<typename E > virtual decaf::util::Iterator<E>*
decaf::util::concurrent::SynchronousQueue< E >::iterator () const
[inline, virtual]`
- 6.668.3.9** `template<typename E > virtual decaf::util::Iterator<E>*
decaf::util::concurrent::SynchronousQueue< E >::iterator () [inline,
virtual]`
- 6.668.3.10** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::offer
(const E & value) throw (decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

Inserts the specified element into this queue, if another thread is waiting to receive it.

Parameters:

value the element to add to the **Queue** (p.2671)

Returns:

true if the element was added to this queue, else false

Exceptions:

NullPointerException if the **Queue** (p.2671) implementation does not allow Null values to be inserted into the **Queue** (p.2671).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

- 6.668.3.11** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::offer
(const E & e, long timeout, const TimeUnit & unit)
throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException, de-
cafe::lang::exceptions::IllegalArgumentException) [inline,
virtual]`

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

Returns:

true if successful, or false if the specified waiting time elapses before a consumer appears.

Exceptions:

InterruptedException

NullPointerException

IllegalArgumentException

6.668.3.12 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::peek (E
&result DECAF_UNUSED) const [inline, virtual]`

6.668.3.13 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::poll (E
& result) [inline, virtual]`

Retrieves and removes the head of this queue, if another thread is currently making an element available.

Parameters:

result a reference to the value where the head of the **Queue** (p. 2671) should be copied to.

Returns:

true if the head of the **Queue** (p. 2671) was copied to the result param or false if no value could be returned.

6.668.3.14 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::poll (E
& result, long long timeout, const TimeUnit & unit) throw (
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

Parameters:

result a reference to the value where the head of the **Queue** (p. 2671) should be copied to.

timeout the time that the method should block if there is no element available to return.

unit the Time Units that the timeout value represents.

Returns:

true if the head of the **Queue** (p. 2671) was copied to the result param or false if no value could be returned.

Referenced by `decaf::util::concurrent::SynchronousQueue< E >::drainTo()`.

6.668.3.15 `template<typename E > virtual void
decaf::util::concurrent::SynchronousQueue< E >::put
(const E & value) throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException, de-
cafe::lang::exceptions::IllegalArgumentException) [inline,
virtual]`

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

Parameters:

value the element to add to the **Queue** (p.2671).

Exceptions:

InterruptedException

NullPointerException

IllegalArgumentException

- 6.668.3.16 `template<typename E > virtual int
decaf::util::concurrent::SynchronousQueue< E
>::remainingCapacity () const [inline, virtual]`
- 6.668.3.17 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::remove
(const E &value DECAF_UNUSED) throw (
lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`
- 6.668.3.18 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E
>::removeAll (const Collection< E > &collection DECAF_UNUSED)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`
- 6.668.3.19 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E
>::retainAll (const Collection< E > &collection DECAF_UNUSED)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`
- 6.668.3.20 `template<typename E > virtual std::size_t
decaf::util::concurrent::SynchronousQueue< E >::size () const [inline,
virtual]`
- 6.668.3.21 `template<typename E > virtual E
decaf::util::concurrent::SynchronousQueue< E >::take ()
throw (decaf::lang::exceptions::InterruptedException) [inline,
virtual]`

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

Returns:

the head of this queue

Exceptions:

InterruptedException

```
6.668.3.22  template<typename E > virtual std::vector<E>
              decaf::util::concurrent::SynchronousQueue< E >::toArray () const
              [inline, virtual]
```

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/SynchronousQueue.h`

6.669 decaf::lang::System Class Reference

```
#include <src/main/decaf/lang/System.h>
```

Public Member Functions

- **System** ()
- virtual **~System** ()

Static Public Member Functions

- static const **util::Map**< std::string, std::string > & **getenv** () throw (lang::Exception)
Enumerates the system environment and returns a map of env variable names to the string values they hold.
- static std::string **getenv** (const std::string &name) throw (lang::Exception)
Reads an environment value from the system and returns it as a string object.
- static void **unsetenv** (const std::string &name) throw (lang::Exception)
Clears a set env value if one is set.
- static void **setenv** (const std::string &name, const std::string &value) throw (lang::Exception)
Sets the specified system property to the value given.
- static long long **currentTimeMillis** ()
Returns the current time in milliseconds.
- static long long **nanoTime** ()
Returns the current value of the most precise available system timer, in nanoseconds.
- static int **availableProcessors** ()
Returns the number of processors available for exection of Decaf Threads.

6.669.1 Constructor & Destructor Documentation

6.669.1.1 decaf::lang::System::System ()

6.669.1.2 virtual decaf::lang::System::~~System () [inline, virtual]

6.669.2 Member Function Documentation

6.669.2.1 static int decaf::lang::System::availableProcessors () [static]

Returns the number of processors available for exection of Decaf Threads. This value may change during a particular execution of a Decaf based application. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

Returns:

the number of available processors.

6.669.2.2 static long long decaf::lang::System::currentTimeMillis () [static]

Returns the current time in milliseconds. Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class Date for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

Returns:

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

**6.669.2.3 static std::string decaf::lang::System::getenv (const std::string & name)
 throw (lang::Exception) [static]**

Reads an environment value from the system and returns it as a string object.

Parameters:

name - the env var to read

Returns:

a string with the value from the var or ""

Exceptions:

an Exception (p. 1574) if an error occurs while reading the Env.

**6.669.2.4 static const util::Map<std::string, std::string>&
 decaf::lang::System::getenv () throw (lang::Exception) [static]**

Enumerates the system environment and returns a map of env variable names to the string values they hold.

Returns:

A Map of all environment variables.

Exceptions:

Exception (p. 1574) if an error occurs

6.669.2.5 static long long decaf::lang::System::nanoTime () [static]

Returns the current value of the most precise available system timer, in nanoseconds. This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

```
long long startTime = System::nanoTime() (p.3147); // ... the code being measured ... long
long estimatedTime = System::nanoTime() (p.3147) - startTime;
```

Returns:

The current value of the system timer, in nanoseconds.

6.669.2.6 static void decaf::lang::System::setenv (const std::string & name, const std::string & value) throw (lang::Exception) [static]

Sets the specified system property to the value given.

Parameters:

name - name of the env val to set
value - value to assign to name

Exceptions:

an Exception (p. 1574) if an error occurs

6.669.2.7 static void decaf::lang::System::unsetenv (const std::string & name) throw (lang::Exception) [static]

Clears a set env value if one is set.

Parameters:

name - the env var to clear

Exceptions:

an Exception (p. 1574) if an error occurs while reading the Env.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**System.h**

6.670 activemq::threads::Task Class Reference

Represents a unit of work that requires one or more iterations to complete.

#include <src/main/activemq/threads/Task.h> Inheritance diagram for activemq::threads::Task:

Public Member Functions

- virtual `~Task()`
- virtual bool `iterate()`=0

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.670.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

Since:

3.0

6.670.2 Constructor & Destructor Documentation

6.670.2.1 virtual `activemq::threads::Task::~~Task()` [inline, virtual]

6.670.3 Member Function Documentation

6.670.3.1 virtual bool `activemq::threads::Task::iterate()` [pure virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns:

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::core::ActiveMQSessionExecutor` (p. 462), `activemq::threads::CompositeTaskRunner` (p. 1093), `activemq::transport::failover::BackupTransportPool` (p. 649), `activemq::transport::failover::CloseTransportsTask` (p. 1023), and `activemq::transport::failover::FailoverTransport` (p. 1618).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

6.671 decaf::util::concurrent::TaskListener Class Reference

```
#include <src/main/decaf/util/concurrent/TaskListener.h>
```

Public Member Functions

- virtual `~TaskListener ()`
- virtual void `onTaskComplete (lang::Runnable *task)=0`
Called when a queued task has completed, the task that finished is passed along for user consumption.
- virtual void `onTaskException (lang::Runnable *task, lang::Exception &ex)=0`
Called when a queued task has thrown an exception while being run.

6.671.1 Constructor & Destructor Documentation

- 6.671.1.1** virtual `decaf::util::concurrent::TaskListener::~~TaskListener ()` [inline, virtual]

6.671.2 Member Function Documentation

- 6.671.2.1** virtual void `decaf::util::concurrent::TaskListener::onTaskComplete (lang::Runnable * task)` [pure virtual]

Called when a queued task has completed, the task that finished is passed along for user consumption.

Parameters:

task Runnable Pointer to the task that finished

- 6.671.2.2** virtual void `decaf::util::concurrent::TaskListener::onTaskException (lang::Runnable * task, lang::Exception & ex)` [pure virtual]

Called when a queued task has thrown an exception while being run. The Callee should assume that this was an unrecoverable exception and that this task is now defunct.

Parameters:

task Runnable Pointer to the task

ex The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TaskListener.h`

6.672 activemq::threads::TaskRunner Class Reference

#include <src/main/activemq/threads/TaskRunner.h> Inheritance diagram for activemq::threads::TaskRunner:

Public Member Functions

- virtual `~TaskRunner ()`
- virtual void **shutdown** (unsigned int timeout)=0
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()=0
Shutdown once the task has finished and the TaskRunner's thread has exited.
- virtual void **wakeup** ()=0
*Signal the **TaskRunner** (p. 3150) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3148) instance will be run until its iterate method has returned false indicating it is done.*

6.672.1 Constructor & Destructor Documentation

6.672.1.1 virtual activemq::threads::TaskRunner::~~TaskRunner () [inline, virtual]

6.672.2 Member Function Documentation

6.672.2.1 virtual void activemq::threads::TaskRunner::shutdown () [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implemented in **activemq::threads::CompositeTaskRunner** (p.1093), and **activemq::threads::DedicatedTaskRunner** (p.1474).

6.672.2.2 virtual void activemq::threads::TaskRunner::shutdown (unsigned int *timeout*) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters:

timeout - Time in Milliseconds to wait for the task to stop.

Implemented in **activemq::threads::CompositeTaskRunner** (p.1094), and **activemq::threads::DedicatedTaskRunner** (p.1474).

6.672.2.3 virtual void activemq::threads::TaskRunner::wakeup () [pure virtual]

Signal the **TaskRunner** (p. 3150) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3148) instance will be run until its iterate method has returned false indicating it is done.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1094), and **activemq::threads::DedicatedTaskRunner** (p. 1474).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

6.673 decaf::net::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

#include <src/main/decaf/net/TcpSocket.h>Inheritance diagram for decaf::net::TcpSocket:

Public Member Functions

- **TcpSocket** () throw (SocketException)
Construct a non-connected socket.
- **TcpSocket** (SocketHandle socketHandle)
Construct a connected or bound socket based on given socket handle.
- virtual ~**TcpSocket** ()
Releases the socket handle but not gracefully shut down the connection.
- **SocketHandle** **getSocketHandle** ()
Gets the handle for the socket.
- void **connect** (const char *host, int port, int timeout) throw (SocketException)
Connects to the specified destination.
- virtual void **connect** (const char *host, int port) throw (SocketException)
Connects to the specified destination.
- virtual bool **isConnected** () const
Indicates whether or not this socket is connected to a destination.
- virtual **io::InputStream** * **getInputStream** ()
Gets the InputStream for this socket.
- virtual **io::OutputStream** * **getOutputStream** ()
Gets the OutputStream for this socket.
- virtual int **getSoLinger** () const throw (SocketException)
Gets the linger time.
- virtual void **setSoLinger** (int linger) throw (SocketException)
Sets the linger time.
- virtual bool **getKeepAlive** () const throw (SocketException)
Gets the keep alive flag.
- virtual void **setKeepAlive** (bool keepAlive) throw (SocketException)
Enables/disables the keep alive flag.
- virtual int **getReceiveBufferSize** () const throw (SocketException)

Gets the receive buffer size.

- virtual void **setReceiveBufferSize** (int size) throw (SocketException)
Sets the receive buffer size.
- virtual bool **getReuseAddress** () const throw (SocketException)
Gets the reuse address flag.
- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
Sets the reuse address flag.
- virtual int **getSendBufferSize** () const throw (SocketException)
Gets the send buffer size.
- virtual void **setSendBufferSize** (int size) throw (SocketException)
Sets the send buffer size.
- virtual int **getSoTimeout** () const throw (SocketException)
Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout) throw (SocketException)
Sets the timeout for socket operations.
- virtual void **close** () throw (decaf::io::IOException)
Closes this object and deallocates the appropriate resources.
- virtual bool **getTcpNoDelay** () const throw (lang::Exception)
Gets the Status of the TCP_NODELAY param for this socket as a Bool.
- virtual void **setTcpNoDelay** (bool value) throw (lang::Exception)
Sets the Status of the TCP_NODELAY param for this socket as a Bool.

Protected Member Functions

- void **checkResult** (apr_status_t value) const throw (SocketException)

6.673.1 Detailed Description

Platform-independent implementation of the socket interface.

6.673.2 Constructor & Destructor Documentation

6.673.2.1 decaf::net::TcpSocket::TcpSocket () throw (SocketException)

Construct a non-connected socket.

Exceptions:

SocketException (p. 2972) thrown on windows if the static initialization call to WSAS-tartup was not successful.

6.673.2.2 decaf::net::TcpSocket::TcpSocket (SocketHandle *socketHandle*)

Construct a connected or bound socket based on given socket handle.

Parameters:

socketHandle a socket handle to wrap in the object

6.673.2.3 virtual decaf::net::TcpSocket::~~TcpSocket () [virtual]

Releases the socket handle but not gracefully shut down the connection.

6.673.3 Member Function Documentation**6.673.3.1 void decaf::net::TcpSocket::checkResult (apr_status_t *value*) const throw (SocketException) [protected]****6.673.3.2 virtual void decaf::net::TcpSocket::close () throw (decaf::io::IOException) [virtual]**

Closes this object and deallocates the appropriate resources.

Exceptions:

IOException

Implements **decaf::io::Closeable** (p. 1019).

6.673.3.3 virtual void decaf::net::TcpSocket::connect (const char * *host*, int *port*) throw (SocketException) [inline, virtual]

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions:

SocketException (p. 2972) Thrown if a failure occurred in the connect.

Implements **decaf::net::Socket** (p. 2965).

6.673.3.4 void decaf::net::TcpSocket::connect (const char * *host*, int *port*, int *timeout*) throw (SocketException)

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

timeout of socket in microseconds

Exceptions:

SocketException (p. 2972) Thrown if a failure occurred in the connect.

6.673.3.5 virtual io::InputStream* decaf::net::TcpSocket::getInputStream ()
[virtual]

Gets the InputStream for this socket.

Returns:

The InputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2966).

6.673.3.6 virtual bool decaf::net::TcpSocket::getKeepAlive () const throw (
SocketException) [virtual]

Gets the keep alive flag.

Returns:

True if keep alive is enabled.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2966).

6.673.3.7 virtual io::OutputStream* decaf::net::TcpSocket::getOutputStream ()
[virtual]

Gets the OutputStream for this socket.

Returns:

the OutputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2966).

6.673.3.8 virtual int decaf::net::TcpSocket::getReceiveBufferSize () const throw (
SocketException) [virtual]

Gets the receive buffer size.

Returns:

the receive buffer size in bytes.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2966).

6.673.3.9 **virtual bool decaf::net::TcpSocket::getReuseAddress () const throw (**
SocketException) [virtual]

Gets the reuse address flag.

Returns:

True if the address can be reused.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2967).

6.673.3.10 **virtual int decaf::net::TcpSocket::getSendBufferSize () const throw (**
SocketException) [virtual]

Gets the send buffer size.

Returns:

the size in bytes of the send buffer.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2967).

6.673.3.11 **SocketHandle decaf::net::TcpSocket::getSocketHandle () [inline]**

Gets the handle for the socket.

Returns:

SocketHabler for this **Socket** (p. 2964), can be NULL

6.673.3.12 **virtual int decaf::net::TcpSocket::getSoLinger () const throw (**
SocketException) [virtual]

Gets the linger time.

Returns:

The linger time in seconds.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2967).

6.673.3.13 `virtual int decaf::net::TcpSocket::getSoTimeout () const throw (SocketException) [virtual]`

Gets the timeout for socket operations.

Returns:

The timeout in milliseconds for socket operations.

Exceptions:

SocketException (p. 2972) Thrown if unable to retrieve the information.

Implements **decaf::net::Socket** (p. 2967).

6.673.3.14 `virtual bool decaf::net::TcpSocket::getTcpNoDelay () const throw (lang::Exception) [virtual]`

Gets the Status of the TCP_NODELAY param for this socket as a Bool.

Returns:

true if TCP_NODELAY is enabled

Exceptions:

Exception

6.673.3.15 `virtual bool decaf::net::TcpSocket::isConnected () const [inline, virtual]`

Indicates whether or not this socket is connected to a destination.

Returns:

true if connected

Implements **decaf::net::Socket** (p. 2968).

6.673.3.16 `virtual void decaf::net::TcpSocket::setKeepAlive (bool keepAlive) throw (SocketException) [virtual]`

Enables/disables the keep alive flag.

Parameters:

keepAlive If true, enables the flag.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements **decaf::net::Socket** (p. 2968).

6.673.3.17 `virtual void decaf::net::TcpSocket::setReceiveBufferSize (int size) throw (SocketException)` [virtual]

Sets the receive buffer size.

Parameters:

size Number of bytes to set the receive buffer to.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2968).

6.673.3.18 `virtual void decaf::net::TcpSocket::setReuseAddress (bool reuse) throw (SocketException)` [virtual]

Sets the reuse address flag.

Parameters:

reuse If true, sets the flag.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2969).

6.673.3.19 `virtual void decaf::net::TcpSocket::setSendBufferSize (int size) throw (SocketException)` [virtual]

Sets the send buffer size.

Parameters:

size The number of bytes to set the send buffer to.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2969).

6.673.3.20 `virtual void decaf::net::TcpSocket::setSoLinger (int linger) throw (SocketException)` [virtual]

Sets the linger time.

Parameters:

linger The linger time in seconds. If 0, linger is off.

Exceptions:

SocketException (p. 2972) if the operation fails.

Implements `decaf::net::Socket` (p. 2969).

6.673.3.21 virtual void decaf::net::TcpSocket::setSoTimeout (int *timeout*) throw (SocketException) [virtual]

Sets the timeout for socket operations.

Parameters:

timeout The timeout in milliseconds for socket operations.

Exceptions:

SocketException (p. 2972) Thrown if unable to set the information.

Implements decaf::net::Socket (p. 2969).

6.673.3.22 virtual void decaf::net::TcpSocket::setTcpNoDelay (bool *value*) throw (lang::Exception) [virtual]

Sets the Status of the TCP_NODELAY param for this socket as a Bool.

Parameters:

value - true if TCP_NODELAY is to be enabled

Exceptions:

Exception

The documentation for this class was generated from the following file:

- src/main/decaf/net/TcpSocket.h

6.674 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1823).

#include <src/main/activemq/transport/tcp/TcpTransport.h> Inheritance diagram for activemq::transport::tcp::TcpTransport:

Public Member Functions

- **TcpTransport** (const **decaf::util::Properties** &properties, const **Pointer**< **Transport** > &next)

Constructor.

- **TcpTransport** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties, const **Pointer**< **Transport** > &next)

Constructor.

- virtual ~**TcpTransport** ()
- virtual void **close** () throw (**decaf::io::IOException**)

Delegates to the superclass and then closes the socket.

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3273) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3273) been shutdown and no longer usable.*

6.674.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1823). The lower level transport should take care of managing stream reads and writes.

6.674.2 Constructor & Destructor Documentation

- #### 6.674.2.1 activemq::transport::tcp::TcpTransport::TcpTransport (const decaf::util::Properties & *properties*, const **Pointer**< **Transport** > & *next*)

Constructor.

Parameters:

properties the configuration properties for this transport
next the next transport in the chain

6.674.2.2 `activemq::transport::tcp::TcpTransport::TcpTransport (const decaf::net::URI & uri, const decaf::util::Properties & properties, const Pointer< Transport > & next)`

Constructor.

Parameters:

uri - The URI containing the host to connect to.
properties the configuration properties for this transport
next the next transport in the chain

6.674.2.3 `virtual activemq::transport::tcp::TcpTransport::~~TcpTransport ()`
 [virtual]

6.674.3 Member Function Documentation

6.674.3.1 `virtual void activemq::transport::tcp::TcpTransport::close () throw (decaf::io::IOException)` [virtual]

Delegates to the superclass and then closes the socket.

Exceptions:

IOException if errors occur.

Reimplemented from `activemq::transport::TransportFilter` (p. 3283).

6.674.3.2 `virtual bool activemq::transport::tcp::TcpTransport::isClosed () const`
 [inline, virtual]

Has the **Transport** (p. 3273) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 3273)

Reimplemented from `activemq::transport::TransportFilter` (p. 3284).

6.674.3.3 `virtual bool activemq::transport::tcp::TcpTransport::isConnected () const`
 [inline, virtual]

Is the **Transport** (p. 3273) Connected to its Broker.

Returns:

true if a connection has been made.

Reimplemented from `activemq::transport::TransportFilter` (p. 3284).

6.674.3.4 `virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant ()` `const [inline, virtual]`

Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 3273) is fault tolerant.

Reimplemented from **activemq::transport::TransportFilter** (p. 3284).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransport.h`

6.675 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 3160).

#include <src/main/activemq/transport/tcp/TcpTransportFactory.h> Inheritance diagram for activemq::transport::tcp::TcpTransportFactory:

Public Member Functions

- virtual **~TcpTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a fully configured **Transport** (p. 3273) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.*

6.675.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 3160).

6.675.2 Constructor & Destructor Documentation

6.675.2.1 virtual
 activemq::transport::tcp::TcpTransportFactory::~~TcpTransportFactory ()
 [inline, virtual]

6.675.3 Member Function Documentation

6.675.3.1 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException)
 [virtual]

Creates a fully configured **Transport** (p.3273) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p.3279).

6.675.3.2 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::createComposite (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException)
 [virtual]

Creates a slimed down **Transport** (p.3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p.3280).

6.675.3.3 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite (const decaf::net::URI & *location*, const Pointer< wireformat::WireFormat > & *wireFormat*, const decaf::util::Properties & *properties*) throw (exceptions::ActiveMQException) [protected, virtual]

Creates a slimed down **Transport** (p.3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 3273).

properties - Properties to apply to the transport.

Returns:

new Pointer to a **TcpTransport** (p. 3160).

Exceptions:

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransportFactory.h**

6.676 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 2674) based **Destination** (p. 1480).

#include <src/main/cms/TemporaryQueue.h> Inheritance diagram for cms::TemporaryQueue:

Public Member Functions

- virtual **~TemporaryQueue** ()
- virtual std::string **getQueueName** () const =0 throw (CMSEException)
Gets the name of this queue.
- virtual void **destroy** ()=0 throw (CMSEException)
*Destroy's the Temporary **Destination** (p. 1480) at the Provider.*

6.676.1 Detailed Description

Defines a Temporary **Queue** (p. 2674) based **Destination** (p. 1480). A **TemporaryQueue** (p. 3166) is a special type of **Queue** (p. 2674) **Destination** (p. 1480) that can only be consumed from the **Connection** (p. 1131) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p. 3166) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1131) that created it.

Since:

1.0

6.676.2 Constructor & Destructor Documentation

6.676.2.1 virtual cms::TemporaryQueue::~~TemporaryQueue () [inline, virtual]

6.676.3 Member Function Documentation

6.676.3.1 virtual void cms::TemporaryQueue::destroy () throw (CMSEException)
 [pure virtual]

Destroy's the Temporary **Destination** (p. 1480) at the Provider.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 525).

6.676.3.2 `virtual std::string cms::TemporaryQueue::getQueueName () const throw (CMSEException) [pure virtual]`

Gets the name of this queue.

Returns:

The queue name.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTempQueue` (p. 526).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryQueue.h`

6.677 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 3215) based **Destination** (p. 1480).

#include <src/main/cms/TemporaryTopic.h> Inheritance diagram for cms::TemporaryTopic:

Public Member Functions

- virtual **~TemporaryTopic** ()
- virtual std::string **getTopicName** () const =0 throw (CMSEException)
Gets the name of this topic.
- virtual void **destroy** ()=0 throw (CMSEException)
*Destroy's the Temporary **Destination** (p. 1480) at the Provider.*

6.677.1 Detailed Description

Defines a Temporary **Topic** (p. 3215) based **Destination** (p. 1480). A **TemporaryTopic** (p. 3168) is a special type of **Topic** (p. 3215) **Destination** (p. 1480) that can only be consumed from the **Connection** (p. 1131) which created it. TemporaryTopics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 3168) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1131) that created it.

Since:

1.0

6.677.2 Constructor & Destructor Documentation

6.677.2.1 virtual cms::TemporaryTopic::~~TemporaryTopic () [inline, virtual]

6.677.3 Member Function Documentation

6.677.3.1 virtual void cms::TemporaryTopic::destroy () throw (CMSEException)
[pure virtual]

Destroy's the Temporary **Destination** (p. 1480) at the Provider.

Exceptions:

CMSEException (p. 1031)

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 550).

6.677.3.2 `virtual std::string cms::TemporaryTopic::getTopicName () const throw (CMSException) [pure virtual]`

Gets the name of this topic.

Returns:

The topic name.

Exceptions:

CMSException (p. 1031) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 551).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryTopic.h`

6.678 cms::TextMessage Class Reference

Interface for a text message.

#include <src/main/cms/TextMessage.h> Inheritance diagram for cms::TextMessage:

Public Member Functions

- virtual `~TextMessage ()`
- virtual `std::string getText () const` `=0` throw (cms::CMSEException)
Gets the message character buffer.
- virtual void `setText (const char *msg)=0` throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void `setText (const std::string &msg)=0` throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets the message contents.

6.678.1 Detailed Description

Interface for a text message. A **TextMessage** (p. 3170) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 3170) is received in Read-Only mode, any attempt to write to the **Message** (p. 2163) will result in a `MessageNotWritableException` being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

Since:

1.0

6.678.2 Constructor & Destructor Documentation

6.678.2.1 virtual cms::TextMessage::~TextMessage () [inline, virtual]

6.678.3 Member Function Documentation

6.678.3.1 virtual std::string cms::TextMessage::getText () const throw (cms::CMSEException) [pure virtual]

Gets the message character buffer.

Returns:

The message character buffer.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 576).

6.678.3.2 `virtual void cms::TextMessage::setText (const std::string & msg) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets the message contents.

Parameters:

msg The message buffer.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode..

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 576).

6.678.3.3 `virtual void cms::TextMessage::setText (const char * msg) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters:

msg The message buffer.

Exceptions:

CMSEException (p. 1031) - if an internal error occurs.

MessageNotWriteableException (p. 2331) - if the message is in read-only mode..

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 576).

The documentation for this class was generated from the following file:

- `src/main/cms/TextMessage.h`

6.679 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 3172)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

Public Member Functions

- virtual **~ThreadFactory** ()
- virtual decaf::lang::Thread * **newThread** (decaf::lang::Runnable *r)=0
Constructs a new Thread.

6.679.1 Detailed Description

public interface **ThreadFactory** (p. 3172) An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p. 3172) { public: Thread* newThread(
Runnable* r ) { return new Thread(r); } }
```

The Executors.defaultThreadFactory() method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since:

1.0

6.679.2 Constructor & Destructor Documentation

6.679.2.1 virtual decaf::util::concurrent::ThreadFactory::~~ThreadFactory ()
[inline, virtual]

6.679.3 Member Function Documentation

6.679.3.1 virtual decaf::lang::Thread* decaf::util::concurrent::ThreadFactory::newThread (decaf::lang::Runnable * r) [pure virtual]

Constructs a new Thread. Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

Parameters:

r A pointer to a Runnable instance to be executed by new Thread instance returned.

Returns:

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadFactory.h`

6.680 decaf::lang::ThreadGroup Class Reference

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

Public Member Functions

- **ThreadGroup** ()
- virtual **~ThreadGroup** ()

6.680.1 Detailed Description

Since:

1.0

6.680.2 Constructor & Destructor Documentation

6.680.2.1 decaf::lang::ThreadGroup::ThreadGroup ()

6.680.2.2 virtual decaf::lang::ThreadGroup::~~ThreadGroup () [virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ThreadGroup.h`

6.681 decaf::util::concurrent::ThreadPool Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

#include <src/main/decaf/util/concurrent/ThreadPool.h> Inheritance diagram for decaf::util::concurrent::ThreadPool:

Public Types

- typedef std::pair< lang::Runnable *, TaskListener * > Task

Public Member Functions

- **ThreadPool** ()
- virtual **~ThreadPool** ()
- virtual void **queueTask** (Task task) throw (lang::Exception)
Queue (p. 2671) a task to be completed by one of the Pooled Threads.
- virtual **Task deQueueTask** () throw (lang::Exception)
DeQueue a task to be completed by one of the Pooled Threads.
- virtual std::size_t **getPoolSize** () const
Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.
- virtual std::size_t **getBacklog** () const
Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.
- virtual void **reserve** (std::size_t size)
Ensures that there is at least the specified number of Threads allocated to the pool.
- virtual std::size_t **getMaxThreads** () const
Get the Max Number of Threads this Pool can contain.
- virtual void **setMaxThreads** (std::size_t maxThreads)
Sets the Max number of threads this pool can contain.
- virtual std::size_t **getBlockSize** () const
Gets the Max number of threads that can be allocated at a time when new threads are needed.
- virtual void **setBlockSize** (std::size_t blockSize)
Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.
- virtual std::size_t **getFreeThreadCount** () const
Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.

- virtual void **onTaskStarted** (**PooledThread** *thread)

Called by a pooled thread when it is about to begin executing a new task.

- virtual void **onTaskCompleted** (**PooledThread** *thread)

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

- virtual void **onTaskException** (**PooledThread** *thread, **lang::Exception** &ex)

*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2518) is now no longer running.*

Static Public Member Functions

- static **ThreadPool** * **getInstance** ()

Return the one and only Thread Pool instance.

Static Public Attributes

- static const size_t **DEFAULT__MAX__POOL__SIZE** = 10
- static const size_t **DEFAULT__MAX__BLOCK__SIZE** = 3

6.681.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks. The Thread Pool has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the **Runnable** interface and one of the worker threads will executing it in its thread context.

6.681.2 Member Typedef Documentation

6.681.2.1 `typedef std::pair<lang::Runnable*, TaskListener*>
decaf::util::concurrent::ThreadPool::Task`

6.681.3 Constructor & Destructor Documentation

6.681.3.1 `decaf::util::concurrent::ThreadPool::ThreadPool ()`

6.681.3.2 `virtual decaf::util::concurrent::ThreadPool::~~ThreadPool ()` [virtual]

6.681.4 Member Function Documentation

6.681.4.1 `virtual Task decaf::util::concurrent::ThreadPool::deQueueTask () throw (
lang::Exception)` [virtual]

DeQueue a task to be completed by one of the Pooled Threads. A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p.3175) don't use this, only the **PooledThread** (p.2518) objects owned by this **ThreadPool** (p.3175).

Returns:

object that derives from Runnable

Exceptions:

ActiveMQException

6.681.4.2 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog ()
const` [inline, virtual]

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

Returns:

number of outstanding tasks.

6.681.4.3 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize ()
const` [inline, virtual]

Gets the Max number of threads that can be allocated at a time when new threads are needed.

Returns:

max Thread Block Size

6.681.4.4 `virtual std::size_t de-
caf::util::concurrent::ThreadPool::getFreeThreadCount ()
const` [inline, virtual]

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable. This value could change immediately after calling as Threads

could finish right after and be available again. This is informational only.

Returns:

total free threads

6.681.4.5 static ThreadPool* decaf::util::concurrent::ThreadPool::getInstance ()
[static]

Return the one and only Thread Pool instance.

Returns:

The Thread Pool Pointer

6.681.4.6 virtual std::size_t decaf::util::concurrent::ThreadPool::getMaxThreads ()
const [inline, virtual]

Get the Max Number of Threads this Pool can contain.

Returns:

max size

6.681.4.7 virtual std::size_t decaf::util::concurrent::ThreadPool::getPoolSize ()
const [inline, virtual]

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

Returns:

integer number of threads in existence.

6.681.4.8 virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted
(PooledThread * *thread*) [virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

Parameters:

thread Pointer the the Pooled Thread that is making this call.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2520).

6.681.4.9 virtual void decaf::util::concurrent::ThreadPool::onTaskException
(PooledThread * *thread*, lang::Exception & *ex*) [virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2518) is now no longer running.

Parameters:

thread Pointer to the Pooled Thread that is making this call
ex The Exception that occurred.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2521).

6.681.4.10 virtual void decaf::util::concurrent::ThreadPool::onTaskStarted (PooledThread * *thread*) [virtual]

Called by a pooled thread when it is about to begin executing a new task. This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

Parameters:

thread Pointer to the Pooled Thread that is making this call

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2521).

6.681.4.11 virtual void decaf::util::concurrent::ThreadPool::queueTask (Task *task*) throw (lang::Exception) [virtual]

Queue (p. 2671) a task to be completed by one of the Pooled Threads. tasks are serviced as soon as a PooledThread (p. 2518) is available to run it.

Parameters:

task object that derives from Runnable

Exceptions:

ActiveMQException

6.681.4.12 virtual void decaf::util::concurrent::ThreadPool::reserve (std::size_t *size*) [virtual]

Ensures that there is at least the specified number of Threads allocated to the pool. If the size is greater than the MAX number of threads in the pool, then only MAX threads are reserved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

Parameters:

size the number of threads to reserve.

6.681.4.13 virtual void decaf::util::concurrent::ThreadPool::setBlockSize (std::size_t *blockSize*) [virtual]

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

Parameters:

blockSize Max Thread Block Size

6.681.4.14 `virtual void decaf::util::concurrent::ThreadPool::setMaxThreads`
`(std::size_t maxThreads)` [virtual]

Sets the Max number of threads this pool can contain. if this value is smaller than the current size of the pool nothing is done.

Parameters:

maxThreads total number of threads that can be pooled

6.681.5 Field Documentation

6.681.5.1 `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3` [static]

6.681.5.2 `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadPool.h`

6.682 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

#include <src/main/decaf/lang/Throwable.h> Inheritance diagram for decaf::lang::Throwable:

Public Member Functions

- **Throwable** () throw ()
- virtual ~**Throwable** () throw ()
- virtual std::string **getMessage** () const =0
Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.
- virtual const std::exception * **getCause** () const =0
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception *cause)=0
Initializes the contained cause exception with the one given.
- virtual void **setMark** (const char *file, const int lineNumber)=0
Adds a file/line number to the stack trace.
- virtual **Throwable** * **clone** () const =0
Clones this exception.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **printStackTrace** () const =0
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const =0
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const =0
Gets the stack trace as one contiguous string.

6.682.1 Detailed Description

This class represents an error that has occurred. All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1574) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.

Throwable (p. 3181) can wrap another **Throwable** (p. 3181) as the cause if the error being thrown. The user can inspect the cause by calling `getCause`, the pointer returned is the property of the **Throwable** (p. 3181) instance and will be deleted when it is deleted or goes out of scope.

Since:

1.0

6.682.2 Constructor & Destructor Documentation

6.682.2.1 `decaf::lang::Throwable::Throwable () throw () [inline]`

6.682.2.2 `virtual decaf::lang::Throwable::~~Throwable () throw () [inline, virtual]`

6.682.3 Member Function Documentation

6.682.3.1 `virtual Throwable* decaf::lang::Throwable::clone () const [pure virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

Copy of this **Exception** (p. 1574) object

Implemented in **activemq::exceptions::ActiveMQException** (p. 306), **activemq::exceptions::BrokerException** (p. 749), **decaf::io::EOFException** (p. 1573), **decaf::io::InterruptedIOException** (p. 1807), **decaf::io::IOException** (p. 1822), **decaf::io::UnsupportedEncodingException** (p. 3302), **decaf::io::UTFDataFormatException** (p. 3355), **decaf::lang::Exception** (p. 1577), **decaf::lang::exceptions::ClassCastException** (p. 1018), **decaf::lang::exceptions::IllegalArgumentException** (p. 1722), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1725), **decaf::lang::exceptions::IllegalStateException** (p. 1728), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1732), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1739), **decaf::lang::exceptions::InterruptedException** (p. 1804), **decaf::lang::exceptions::InvalidStateException** (p. 1819), **decaf::lang::exceptions::NoSuchElementException** (p. 2425), **decaf::lang::exceptions::NullPointerException** (p. 2431), **decaf::lang::exceptions::NumberFormatException** (p. 2437), **decaf::lang::exceptions::RuntimeException** (p. 2821), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3305), **decaf::net::BindException** (p. 723), **decaf::net::ConnectException** (p. 1130), **decaf::net::HttpRetryException** (p. 1719), **decaf::net::MalformedURLException** (p. 2093), **decaf::net::NoRouteToHostException** (p. 2419), **decaf::net::PortUnreachableException** (p. 2524), **decaf::net::ProtocolException** (p. 2669), **decaf::net::SocketException** (p. 2973), **decaf::net::SocketTimeoutException** (p. 2992), **decaf::net::UnknownHostException** (p. 3296), **decaf::net::UnknownServiceException** (p. 3299), **decaf::net::URISyntaxException** (p. 3337), **decaf::nio::BufferOverflowException** (p. 836), **decaf::nio::BufferUnderflowException** (p. 839), **decaf::nio::InvalidMarkException** (p. 1815), **decaf::nio::ReadOnlyBufferException** (p. 2687), **decaf::util::concurrent::BrokenBarrierException** (p. 744),

decaf::util::concurrent::CancellationException	(p. 967),	de-
caf::util::concurrent::ExecutionException	(p. 1607),	de-
caf::util::concurrent::RejectedExecutionException	(p. 2701),	de-
caf::util::concurrent::TimeoutException	(p. 3187).	

6.682.3.2 virtual const std::exception* decaf::lang::Throwable::getCause () const [pure virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1577).

6.682.3.3 virtual std::string decaf::lang::Throwable::getMessage () const [pure virtual]

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.

Returns:

string errors message

Implemented in **decaf::lang::Exception** (p. 1578).

6.682.3.4 virtual std::vector< std::pair< std::string, int> > decaf::lang::Throwable::getStackTrace () const [pure virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns:

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1578).

6.682.3.5 virtual std::string decaf::lang::Throwable::getStackTraceString () const [pure virtual]

Gets the stack trace as one contiguous string.

Returns:

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1578).

6.682.3.6 `virtual void decaf::lang::Throwable::initCause (const std::exception *
cause)` [pure virtual]

Initializes the contained cause exception with the one given. A copy is made to avoid ownership issues.

Parameters:

cause The exception that was the cause of this one.

Implemented in `decaf::lang::Exception` (p. 1578).

6.682.3.7 `virtual void decaf::lang::Throwable::printStackTrace (std::ostream &
stream) const` [pure virtual]

Prints the stack trace to the given output stream.

Parameters:

stream the target output stream.

Implemented in `decaf::lang::Exception` (p. 1579).

6.682.3.8 `virtual void decaf::lang::Throwable::printStackTrace () const` [pure virtual]

Prints the stack trace to std::err.

Implemented in `decaf::lang::Exception` (p. 1579).

6.682.3.9 `virtual void decaf::lang::Throwable::setMark (const char * file, const int
lineNumber)` [pure virtual]

Adds a file/line number to the stack trace.

Parameters:

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

Implemented in `decaf::lang::Exception` (p. 1579).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Throwable.h`

6.683 decaf::util::concurrent::TimeoutException Class Reference

#include <src/main/decaf/util/concurrent/TimeoutException.h> Inheritance diagram for decaf::util::concurrent::TimeoutException:

Public Member Functions

- **TimeoutException** () throw ()
Default Constructor.
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **TimeoutException** (const **TimeoutException** &ex) throw ()
Copy Constructor.
- **TimeoutException** (const std::exception *cause) throw ()
Constructor.
- **TimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **TimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **TimeoutException** * clone () const
Clones this exception.
- virtual ~**TimeoutException** () throw ()

6.683.1 Constructor & Destructor Documentation

6.683.1.1 decaf::util::concurrent::TimeoutException::TimeoutException () throw () [inline]

Default Constructor.

6.683.1.2 decaf::util::concurrent::TimeoutException::TimeoutException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.683.1.3 decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The exception to copy from.

References decaf::lang::Exception::Exception().

6.683.1.4 decaf::util::concurrent::TimeoutException::TimeoutException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.683.1.5 decaf::util::concurrent::TimeoutException::TimeoutException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The string message to report

... list of primitives that are formatted into the message

6.683.1.6 decaf::util::concurrent::TimeoutException::TimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The string message to report

... list of primitives that are formatted into the message

6.683.1.7 virtual decaf::util::concurrent::TimeoutException::~~TimeoutException ()
throw () [inline, virtual]

6.683.2 Member Function Documentation

6.683.2.1 virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new **TimeoutException** (p.3185) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1577).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TimeoutException.h**

6.684 decaf::util::Timer Class Reference

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

Public Member Functions

- **Timer** ()
- virtual **~Timer** ()
- void **cancel** ()
Terminates this timer, discarding any currently scheduled tasks.
- std::size_t **purge** ()
Removes all canceled tasks from this timer's task queue.
- void **schedule** (**TimerTask** *task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution after the specified delay.
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution after the specified delay.
- void **schedule** (**TimerTask** *task, const **Date** &time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution at the specified time.
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for execution at the specified time.
- void **schedule** (**TimerTask** *task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

- void **schedule** (**TimerTask** *task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.
- void **scheduleAtFixedRate** (**TimerTask** *task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.
- void **scheduleAtFixedRate** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.
- void **scheduleAtFixedRate** (**TimerTask** *task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.
- void **scheduleAtFixedRate** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

6.684.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 3188) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 3188) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 3188) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the wait(long) method.

Since:

1.0

6.684.2 Constructor & Destructor Documentation

6.684.2.1 `decaf::util::Timer::Timer ()`

6.684.2.2 `virtual decaf::util::Timer::~~Timer ()` [virtual]

6.684.3 Member Function Documentation

6.684.3.1 `void decaf::util::Timer::cancel ()`

Terminates this timer, discarding any currently scheduled tasks. Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the run method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

6.684.3.2 `std::size_t decaf::util::Timer::purge ()`

Removes all canceled tasks from this timer's task queue. Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p. 3188) to destroy the **TimerTask** (p. 3199) pointer it was originally given, the caller should ensure that they no longer have any references to TimerTasks that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to $n + c \log n$, where n is the number of tasks in the queue and c is the number of canceled tasks.

This method can be called on a **Timer** (p. 3188) object that has no scheduled tasks without error.

Returns:

the number of tasks removed from the queue.

6.684.3.3 `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, const Date & firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity),

subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters:

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.4 `void decaf::util::Timer::schedule (TimerTask * task,
const Date & firstTime, long long period) throw
(decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3199) pointer is considered to be owned by the **Timer** (p. 3188) class once it has been scheduled, the **Timer** (p. 3188) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3188) itself is cancelled. A **TimerTask** (p. 3199) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3188) and the caller should ensure that the **TimerTask** (p. 3199) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3199) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters:

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.5 void decaf::util::Timer::schedule (const decaf::lang::Pointer<
TimerTask > & task, long long delay, long long period)
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters:

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.6 void decaf::util::Timer::schedule (TimerTask * task, long long delay,
long long period) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals separated by the specified

period.

The **TimerTask** (p. 3199) pointer is considered to be owned by the **Timer** (p. 3188) class once it has been scheduled, the **Timer** (p. 3188) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3188) itself is cancelled. A **TimerTask** (p. 3199) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3188) and the caller should ensure that the **TimerTask** (p. 3199) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3199) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters:

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.7 `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, const Date & time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution at the specified time. If the time is in the past, the task is scheduled for immediate execution.

Parameters:

task - task to be scheduled.

time - time at which task is to be executed.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.8 `void decaf::util::Timer::schedule (TimerTask * task, const Date & time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution at the specified time. If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 3199) pointer is considered to be owned by the **Timer** (p. 3188) class once it has been scheduled, the **Timer** (p. 3188) will destroy its TimerTask's once they have been cancelled or the **Timer** (p. 3188) itself is cancelled. A **TimerTask** (p. 3199) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3188) and the caller should ensure that the **TimerTask** (p. 3199) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3199) instance are planned.

Parameters:

task - task to be scheduled.

time - time at which task is to be executed.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.9 `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution after the specified delay.

Parameters:

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, or timer was cancelled.

6.684.3.10 void decaf::util::Timer::schedule (TimerTask * *task*, long long *delay*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for execution after the specified delay. The **TimerTask** (p. 3199) pointer is considered to be owned by the **Timer** (p. 3188) class once it has been scheduled, the **Timer** (p. 3188) will destroy its TimerTask's once they have been cancelled or the **Timer** (p. 3188) itself is cancelled. A **TimerTask** (p. 3199) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3188) and the caller should ensure that the **TimerTask** (p. 3199) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3199) instance are planned.

Parameters:

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, or timer was cancelled.

6.684.3.11 void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & *task*, const Date & *firstTime*, long long *period*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters:

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.12 void decaf::util::Timer::scheduleAtFixedRate (TimerTask
* task, const Date & firstTime, long long period)
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time. Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3199) pointer is considered to be owned by the **Timer** (p. 3188) class once it has been scheduled, the **Timer** (p. 3188) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3188) itself is cancelled. A **TimerTask** (p. 3199) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3188) and the caller should ensure that the **TimerTask** (p. 3199) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3199) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters:

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.13 `void decaf::util::Timer::scheduleAtFixedRate (const decaf::lang::Pointer< TimerTask > & task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters:

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.14 `void decaf::util::Timer::scheduleAtFixedRate (TimerTask * task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay. Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3199) pointer is considered to be owned by the **Timer** (p. 3188) class once it has been scheduled, the **Timer** (p. 3188) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3188) itself is cancelled. A **TimerTask** (p. 3199) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3188) and the caller should ensure that the **TimerTask** (p. 3199) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3199) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters:

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions:

NullPointerException - if the **TimerTask** (p. 3199) value is Null.

IllegalArgumentException - if delay is negative, or delay + `System.currentTimeMillis()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Timer.h`

6.685 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3188).

`#include <src/main/decaf/util/TimerTask.h>`Inheritance diagram for decaf::util::TimerTask:

Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()
- bool **cancel** ()
Cancels this timer task.
- long long **scheduledExecutionTime** () const
Returns the scheduled execution time of the most recent actual execution of this task.

Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

Friends

- class **Timer**
- class **TimerImpl**
- class **decaf::internal::util::TimerTaskHeap**

6.685.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3188).

Since:

1.0

6.685.2 Constructor & Destructor Documentation

6.685.2.1 `decaf::util::TimerTask::TimerTask ()`

6.685.2.2 `virtual decaf::util::TimerTask::~~TimerTask ()` [inline, virtual]

6.685.3 Member Function Documentation

6.685.3.1 `bool decaf::util::TimerTask::cancel ()`

Cancels this timer task. If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

Returns:

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

6.685.3.2 `long long decaf::util::TimerTask::getWhen () const` [protected]

6.685.3.3 `bool decaf::util::TimerTask::isScheduled () const` [protected]

6.685.3.4 `long long decaf::util::TimerTask::scheduledExecutionTime () const`

Returns the scheduled execution time of the most recent actual execution of this task. (If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 2816) { if( System::currentTimeMillis() - scheduledExecutionTime() (p. 3200)
>= MAX_TARDINESS) return; // Too late; skip this execution. // Perform the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

Returns:

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1469). The return value is undefined if the task has yet to commence its first execution.

6.685.3.5 void decaf::util::TimerTask::setScheduledTime (long long *time*)
[protected]

6.685.4 Friends And Related Function Documentation

6.685.4.1 friend class decaf::internal::util::TimerTaskHeap [friend]

6.685.4.2 friend class Timer [friend]

6.685.4.3 friend class TimerImpl [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**TimerTask.h**

6.686 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```

Public Member Functions

- **TimerTaskHeap** ()
- virtual **~TimerTaskHeap** ()
- **Pointer**< **TimerTask** > **peek** ()

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

- bool **isEmpty** () const
- std::size_t **size** () const
- void **insert** (const **Pointer**< **TimerTask** > &task)

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

- void **remove** (std::size_t pos)

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

- void **reset** ()

Clear all contents from the heap.

- void **adjustMinimum** ()

Resorts the heap starting at the top.

- std::size_t **deleteIfCancelled** ()

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

- std::size_t **find** (const **Pointer**< **TimerTask** > &task) const

Searches the heap for the specified TimerTask element and returns its position in the heap.

6.686.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Since:

1.0

6.686.2 Constructor & Destructor Documentation

6.686.2.1 `decaf::internal::util::TimerTaskHeap::TimerTaskHeap ()`

6.686.2.2 `virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap ()`
[virtual]

6.686.3 Member Function Documentation

6.686.3.1 `void decaf::internal::util::TimerTaskHeap::adjustMinimum ()`

Resorts the heap starting at the top.

6.686.3.2 `std::size_t decaf::internal::util::TimerTaskHeap::deleteIfCancelled ()`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

Returns:

the number of task that were removed from the heap because they were cancelled.

6.686.3.3 `std::size_t decaf::internal::util::TimerTaskHeap::find (const Pointer< TimerTask > & task) const`

Searches the heap for the specified TimerTask element and returns its position in the heap. Returns the unsigned equivalent of -1 if the element is not found.

Returns:

the position in the Heap where the Task is stored, or npos.

6.686.3.4 `void decaf::internal::util::TimerTaskHeap::insert (const Pointer< TimerTask > & task)`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

Parameters:

task The TimerTask to insert into the heap.

6.686.3.5 `bool decaf::internal::util::TimerTaskHeap::isEmpty () const`

Returns:

true if the heap is empty.

6.686.3.6 `Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ()`

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

Returns:

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

6.686.3.7 `void decaf::internal::util::TimerTaskHeap::remove (std::size_t pos)`

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

Parameters:

pos The position at which to remove the TimerTask and begin a resort of the heap.

6.686.3.8 `void decaf::internal::util::TimerTaskHeap::reset ()`

Clear all contents from the heap.

6.686.3.9 `std::size_t decaf::internal::util::TimerTaskHeap::size () const`**Returns:**

the size of the heap.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/TimerTaskHeap.h`

6.687 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 3205) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

#include <src/main/decaf/util/concurrent/TimeUnit.h> Inheritance diagram for decaf::util::concurrent::TimeUnit:

Public Member Functions

- virtual **~TimeUnit** ()
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const
Convert the given time duration in the given unit to this unit.
- long long **toNanos** (long long duration) const
Equivalent to `NANOSECONDS.convert(duration, this)`.
- long long **toMicros** (long long duration) const
Equivalent to `MICROSECONDS.convert(duration, this)`.
- long long **toMillis** (long long duration) const
Equivalent to `MILLISECONDS.convert(duration, this)`.
- long long **toSeconds** (long long duration) const
Equivalent to `SECONDS.convert(duration, this)`.
- long long **toMinutes** (long long duration) const
Equivalent to `MINUTES.convert(duration, this)`.
- long long **toHours** (long long duration) const
Equivalent to `HOURS.convert(duration, this)`.
- long long **toDays** (long long duration) const
Equivalent to `DAYS.convert(duration, this)`.
- void **timedWait** (**Synchronizable** *obj, long long timeout) const throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)
Perform a timed `Object.wait` using this time unit.
- void **timedJoin** (decaf::lang::Thread *thread, long long timeout) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)
Perform a timed `Thread.join` using this time unit.
- void **sleep** (long long timeout) const throw (decaf::lang::exceptions::InterruptedException)
Perform a `Thread.sleep` using this unit.

- virtual std::string **toString** () const
*Converts the **TimeUnit** (p. 3205) type to the Name of the **TimeUnit** (p. 3205).*
- virtual int **compareTo** (const **TimeUnit** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **TimeUnit** &value) const
- virtual bool **operator==** (const **TimeUnit** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **TimeUnit** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw (decaf::lang::exceptions::IllegalArgumentException)
*Returns the **TimeUnit** (p. 3205) constant of this type with the specified name.*

Static Public Attributes

- static const **TimeUnit** NANoseconds
*The Actual **TimeUnit** (p. 3205) enumerations.*
- static const **TimeUnit** MICROseconds
- static const **TimeUnit** MILLIseconds
- static const **TimeUnit** SECONDS
- static const **TimeUnit** MINUTES
- static const **TimeUnit** HOURS
- static const **TimeUnit** DAYS
- static const **TimeUnit** *const values []
*The An Array of **TimeUnit** (p. 3205) Instances.*

Protected Member Functions

- **TimeUnit** (int index, const std::string &name)
Hidden Constructor, this class can not be instantiated directly.

6.687.1 Detailed Description

A **TimeUnit** (p. 3205) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A **TimeUnit** (p. 3205) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is

defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 3205) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

Lock (p. 2023) lock = ...; if (lock.tryLock(50, **TimeUnit.MILLISECONDS** (p. 3213))) ...

while this code will timeout in 50 seconds:

Lock (p. 2023) lock = ...; if (lock.tryLock(50, **TimeUnit.SECONDS** (p. 3214))) ...

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 3205).

6.687.2 Constructor & Destructor Documentation

6.687.2.1 decaf::util::concurrent::TimeUnit::TimeUnit (int *index*, const std::string & *name*) [protected]

Hidden Constructor, this class can not be instantiated directly.

Parameters:

index - Index into the Time Unit set.

name - Name of the unit type being represented.

6.687.2.2 virtual decaf::util::concurrent::TimeUnit::~~TimeUnit () [inline, virtual]

6.687.3 Member Function Documentation

6.687.3.1 virtual int decaf::util::concurrent::TimeUnit::compareTo (const TimeUnit & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation $\text{sgn}(\text{expression})$ designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure $\text{sgn}(x.\text{compareTo}(y)) == -\text{sgn}(y.\text{compareTo}(x))$ for all x and y . (This implies that $x.\text{compareTo}(y)$ must throw an exception iff $y.\text{compareTo}(x)$ throws an exception.)

The implementor must also ensure that the relation is transitive: $(x.\text{compareTo}(y) > 0 \ \&\& \ y.\text{compareTo}(z) > 0)$ implies $x.\text{compareTo}(z) > 0$.

Finally, the implementor must ensure that $x.\text{compareTo}(y) == 0$ implies that $\text{sgn}(x.\text{compareTo}(z)) == \text{sgn}(y.\text{compareTo}(z))$, for all z .

It is strongly recommended, but not strictly required that $(x.\text{compareTo}(y) == 0) == (x.\text{equals}(y))$. Generally speaking, any class that implements the Comparable interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.687.3.2 long long decaf::util::concurrent::TimeUnit::convert (long long *sourceDuration*, const TimeUnit & *sourceUnit*) const

Convert the given time duration in the given unit to this unit. Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to Long.MIN_VALUE if negative or Long.MAX_VALUE if positive.

For example, to convert 10 minutes to milliseconds, use: TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES (p. 3213))

Parameters:

sourceDuration - Duration value to convert.

sourceUnit - Unit type of the source duration.

Returns:

the converted duration in this unit, or Long.MIN_VALUE if conversion would negatively overflow, or Long.MAX_VALUE if it would positively overflow.

6.687.3.3 virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & *value*) const [virtual]**Returns:**

true if this value is considered equal to the passed value.

6.687.3.4 virtual bool decaf::util::concurrent::TimeUnit::operator< (const TimeUnit & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.687.3.5 virtual bool decaf::util::concurrent::TimeUnit::operator==(const TimeUnit & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.687.3.6 void decaf::util::concurrent::TimeUnit::sleep (long long *timeout*) const throw (decaf::lang::exceptions::InterruptedException)

Perform a Thread.sleep using this unit. This is a convenience method that converts time arguments into the form required by the Thread.sleep method.

Parameters:

timeout the minimum time to sleep

See also:

Thread::sleep

6.687.3.7 void decaf::util::concurrent::TimeUnit::timedJoin (decaf::lang::Thread * *thread*, long long *timeout*) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)

Perform a timed Thread.join using this time unit. This is a convenience method that converts time arguments into the form required by the Thread.join method.

Parameters:

thread the thread to wait for

timeout the maximum time to wait

Exceptions:

InterruptedException if interrupted while waiting.

NullPointerException if the thread object is null.

See also:

Thread::join(long long, long long)

6.687.3.8 `void decaf::util::concurrent::TimeUnit::timedWait`
(`Synchronizable * obj`, `long long timeout`) `const`
`throw (decaf::lang::exceptions::InterruptedException,`
`decaf::lang::exceptions::NullPointerException)`

Perform a timed `Object.wait` using this time unit. This is a convenience method that converts timeout arguments into the form required by the `Object.wait` method.

For example, you could implement a blocking poll method (see **BlockingQueue.poll** (p. ??)) using:

```
Object poll( long long timeout, const TimeUnit& unit )
    throw( InterruptedException ) {

    while( empty ) {
        unit.timedWait(this, timeout);
        ...
    }
}
```

Parameters:

obj the object to wait on

timeout the maximum time to wait.

Exceptions:

InterruptedException if interrupted while waiting.

NullPointerException if the **Synchronizable** (p. 3122) object is null.

See also:

`Synchronizable::wait(long long, long long)`

6.687.3.9 `long long decaf::util::concurrent::TimeUnit::toDays` (`long long duration`)
`const [inline]`

Equivalent to `DAYS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

`convert` (p. 3208)

6.687.3.10 `long long decaf::util::concurrent::TimeUnit::toHours (long long duration) const [inline]`

Equivalent to `HOURS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

`convert` (p. 3208)

6.687.3.11 `long long decaf::util::concurrent::TimeUnit::toMicros (long long duration) const [inline]`

Equivalent to `MICROSECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also:

`convert` (p. 3208)

6.687.3.12 `long long decaf::util::concurrent::TimeUnit::toMillis (long long duration) const [inline]`

Equivalent to `MILLISECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also:

`convert` (p. 3208)

6.687.3.13 `long long decaf::util::concurrent::TimeUnit::toMinutes (long long duration) const [inline]`

Equivalent to `MINUTES.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

`convert` (p. 3208)

6.687.3.14 `long long decaf::util::concurrent::TimeUnit::toNanos (long long duration) const [inline]`

Equivalent to `NANOSECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also:

`convert` (p. 3208)

6.687.3.15 `long long decaf::util::concurrent::TimeUnit::toSeconds (long long duration) const [inline]`

Equivalent to `SECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

`convert` (p. 3208)

6.687.3.16 `virtual std::string decaf::util::concurrent::TimeUnit::toString () const`
[virtual]

Converts the **TimeUnit** (p. 3205) type to the Name of the **TimeUnit** (p. 3205).

Returns:

String name of the **TimeUnit** (p. 3205)

6.687.3.17 `static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf`
(const std::string & *name*) throw (de-
caf::lang::exceptions::IllegalArgumentException)
[static]

Returns the **TimeUnit** (p. 3205) constant of this type with the specified name. The string must match exactly an identifier used to declare an **TimeUnit** (p. 3205) constant in this type. (Extraaneous whitespace characters are not permitted.)

Parameters:

name The Name of the **TimeUnit** (p. 3205) constant to be returned.

Returns:

A constant reference to the **TimeUnit** (p. 3205) Constant with the given name.

Exceptions:

IllegalArgumentException if this enum type has no constant with the specified name

6.687.4 Field Documentation

6.687.4.1 `const TimeUnit decaf::util::concurrent::TimeUnit::DAYS` [static]

6.687.4.2 `const TimeUnit decaf::util::concurrent::TimeUnit::HOURS` [static]

6.687.4.3 `const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS`
[static]

6.687.4.4 `const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS`
[static]

6.687.4.5 `const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES` [static]

6.687.4.6 `const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS`
[static]

The Actual **TimeUnit** (p. 3205) enumerations.

6.687.4.7 `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS` `[static]`

6.687.4.8 `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`
`[static]`

The An Array of **TimeUnit** (p. 3205) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

6.688 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

#include <src/main/cms/Topic.h> Inheritance diagram for cms::Topic:

Public Member Functions

- virtual `~Topic ()`
- virtual `std::string getTopicName () const =0 throw (CMSEException)`
Gets the name of this topic.

6.688.1 Detailed Description

An interface encapsulating a provider-specific topic name. A **Topic** (p.3215) is a Publish / Subscribe type **Destination** (p.1480). All Messages sent to a **Topic** (p.3215) are broadcast to all Subscribers of that **Topic** (p.3215) unless the Subscriber defines a **Message** (p.2163) selector that filters out that **Message** (p.2163).

Since:

1.0

6.688.2 Constructor & Destructor Documentation

6.688.2.1 virtual `cms::Topic::~~Topic ()` [inline, virtual]

6.688.3 Member Function Documentation

6.688.3.1 virtual `std::string cms::Topic::getTopicName () const throw (CMSEException)` [pure virtual]

Gets the name of this topic.

Returns:

The topic name.

Exceptions:

CMSEException (p. 1031) - If an internal error occurs.

Implemented in `activemq::commands::ActiveMQTopic` (p.601).

The documentation for this class was generated from the following file:

- `src/main/cms/Topic.h`

6.689 activemq::state::Tracked Class Reference

#include <src/main/activemq/state/Tracked.h> Inheritance diagram for activemq::state::Tracked:

Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual **~Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

6.689.1 Constructor & Destructor Documentation

6.689.1.1 **activemq::state::Tracked::Tracked** () [inline]

6.689.1.2 **activemq::state::Tracked::Tracked** (const **Pointer**< **decaf::lang::Runnable** > & *runnable*)

6.689.1.3 virtual **activemq::state::Tracked::~~Tracked** () [inline, virtual]

6.689.2 Member Function Documentation

6.689.2.1 bool **activemq::state::Tracked::isWaitingForResponse** () const [inline]

6.689.2.2 void **activemq::state::Tracked::onResponse** ()

The documentation for this class was generated from the following file:

- src/main/activemq/state/**Tracked.h**

6.690 activemq::commands::TransactionId Class Reference

#include <src/main/activemq/commands/TransactionId.h> Inheritance diagram for activemq::commands::TransactionId:

Public Types

- typedef decaf::lang::PointerComparator< TransactionId > COMPARATOR

Public Member Functions

- TransactionId ()
- TransactionId (const TransactionId &other)
- virtual ~TransactionId ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual TransactionId * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const DataStructure *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
Returns a string containing the information for this DataStructure (p. 1461) such as its type and value of its elements.
- virtual bool **equals** (const DataStructure *value) const
Compares the DataStructure (p. 1461) passed in to this one, and returns if they are equivalent.
- virtual int **compareTo** (const TransactionId &value) const
- virtual bool **equals** (const TransactionId &value) const
- virtual bool **operator==** (const TransactionId &value) const
- virtual bool **operator<** (const TransactionId &value) const
- TransactionId & **operator=** (const TransactionId &other)

Static Public Attributes

- static const unsigned char ID_TRANSACTIONID = 0

6.690.1 Member Typedef Documentation

6.690.1.1 typedef decaf::lang::PointerComparator<TransactionId>
activemq::commands::TransactionId::COMPARATOR

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1994), and **activemq::commands::XATransactionId** (p. 3411).

6.690.2 Constructor & Destructor Documentation

6.690.2.1 `activemq::commands::TransactionId::TransactionId ()`

6.690.2.2 `activemq::commands::TransactionId::TransactionId (const TransactionId & other)`

6.690.2.3 `virtual activemq::commands::TransactionId::~~TransactionId ()` [virtual]

6.690.3 Member Function Documentation

6.690.3.1 `virtual TransactionId* activemq::commands::TransactionId::cloneDataStructure ()`
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

Reimplemented in `activemq::commands::LocalTransactionId` (p. 1994), and `activemq::commands::XATransactionId` (p. 3411).

6.690.3.2 `virtual int activemq::commands::TransactionId::compareTo (const TransactionId & value)` const [virtual]

6.690.3.3 `virtual void activemq::commands::TransactionId::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1462).

Reimplemented in `activemq::commands::LocalTransactionId` (p. 1994), and `activemq::commands::XATransactionId` (p. 3411).

6.690.3.4 `virtual bool activemq::commands::TransactionId::equals (const TransactionId & value)` const [virtual]

6.690.3.5 `virtual bool activemq::commands::TransactionId::equals (const DataStructure * value)` const [virtual]

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1463).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1995), and **activemq::commands::XATransactionId** (p. 3412).

6.690.3.6 virtual unsigned char activemq::commands::TransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1995), and **activemq::commands::XATransactionId** (p. 3412).

6.690.3.7 virtual bool activemq::commands::TransactionId::operator< (const TransactionId & value) const [virtual]

6.690.3.8 TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & other)

6.690.3.9 virtual bool activemq::commands::TransactionId::operator== (const TransactionId & value) const [virtual]

6.690.3.10 virtual std::string activemq::commands::TransactionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 718).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1996), and **activemq::commands::XATransactionId** (p. 3413).

6.690.4 Field Documentation

6.690.4.1 const unsigned char activemq::commands::TransactionId::ID_TRANSACTIONID = 0 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionId.h**

6.691 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3220).

#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.691.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3220). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.691.2 Constructor & Destructor Documentation

6.691.2.1 `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.691.2.2 `virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

6.691.3 Member Function Documentation

6.691.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2010), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3420).

6.691.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2011), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3421).

6.691.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2011), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3421).

6.691.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2011), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3421).

6.691.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2012), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3422).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h`

6.692 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3224).

#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.692.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3224). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.692.2 Constructor & Destructor Documentation

6.692.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.692.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

6.692.3 Member Function Documentation

6.692.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 1998), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3428).

6.692.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 1999), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3429).

6.692.3.3 `virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 1999), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3429).

6.692.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1999), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3429).

6.692.3.5 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2000), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3430).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h`

6.693 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3228).

#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.693.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3228). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.693.2 Constructor & Destructor Documentation

6.693.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.693.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

6.693.3 Member Function Documentation

6.693.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2006), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3424).

6.693.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2007), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3425).

6.693.3.3 `virtual int activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2007), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3425).

6.693.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2007), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3425).

6.693.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2008), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3426).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h`

6.694 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3232).

#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.694.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3232). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.694.2 Constructor & Destructor Documentation

6.694.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.694.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

6.694.3 Member Function Documentation

6.694.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2014), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3432).

6.694.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2015), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3433).

6.694.3.3 `virtual int activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2015), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3433).

6.694.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2015), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3433).

6.694.3.5 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions:

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2016), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3434).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h`

6.695 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3236).

#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.695.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3236). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.695.2 Constructor & Destructor Documentation

6.695.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMarshaller()` [inline]

6.695.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~~TransactionIdMarshaller()` [inline, virtual]

6.695.3 Member Function Documentation

6.695.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2002), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3416).

6.695.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2003), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3417).

6.695.3.3 `virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2003), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3417).

6.695.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

6.695 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference 3243

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2003), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3417).

```
6.695.3.5 virtual void ac-
    tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal
    (OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2004), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3418).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h`

6.696 activemq::commands::TransactionInfo Class Reference

#include <src/main/activemq/commands/TransactionInfo.h> Inheritance diagram for activemq::commands::TransactionInfo:

Public Member Functions

- **TransactionInfo** ()
- virtual **~TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **TransactionInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONINFO** = 7

Protected Member Functions

- **TransactionInfo** (const **TransactionInfo** &)
- **TransactionInfo** & **operator=** (const **TransactionInfo** &)

Protected Attributes

- `Pointer< ConnectionId > connectionId`
- `Pointer< TransactionId > transactionId`
- unsigned char `type`

6.696.1 Constructor & Destructor Documentation

6.696.1.1 `activemq::commands::TransactionInfo::TransactionInfo (const TransactionInfo &) [inline, protected]`

6.696.1.2 `activemq::commands::TransactionInfo::TransactionInfo ()`

6.696.1.3 `virtual activemq::commands::TransactionInfo::~~TransactionInfo () [virtual]`

6.696.2 Member Function Documentation

6.696.2.1 `virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.696.2.2 `virtual void activemq::commands::TransactionInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.696.2.3 `virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

- 6.696.2.4** `virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ()`
[virtual]
- 6.696.2.5** `virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ()`
const [virtual]
- 6.696.2.6** `virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Implements **activemq::commands::DataStructure** (p. 1464).

- 6.696.2.7** `virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ()`
[virtual]
- 6.696.2.8** `virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ()`
const [virtual]
- 6.696.2.9** `virtual unsigned char activemq::commands::TransactionInfo::getType ()`
const [virtual]
- 6.696.2.10** `virtual bool activemq::commands::TransactionInfo::isTransactionInfo ()`
const [inline, virtual]

Returns:

an answer of true to the **isTransactionInfo()** (p. 3242) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

- 6.696.2.11 TransactionInfo& activemq::commands::TransactionInfo::operator=(const TransactionInfo &) [inline, protected]
- 6.696.2.12 virtual void activemq::commands::TransactionInfo::setConnectionId(const Pointer< ConnectionId > & *connectionId*) [virtual]
- 6.696.2.13 virtual void activemq::commands::TransactionInfo::setTransactionId(const Pointer< TransactionId > & *transactionId*) [virtual]
- 6.696.2.14 virtual void activemq::commands::TransactionInfo::setType (unsigned char *type*) [virtual]
- 6.696.2.15 virtual std::string activemq::commands::TransactionInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

- 6.696.2.16 virtual Pointer<Command> activemq::commands::TransactionInfo::visit(activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1067).

6.696.3 Field Documentation

- 6.696.3.1 Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId [protected]
- 6.696.3.2 const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7 [static]
- 6.696.3.3 Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId [protected]
- 6.696.3.4 unsigned char activemq::commands::TransactionInfo::type [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionInfo.h`

6.697 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3245).

#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.697.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3245). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.697.2 Constructor & Destructor Documentation

6.697.2.1 `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.697.2.2 `virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.697.3 Member Function Documentation

6.697.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.697.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.697.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 665).

6.697.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 666).

6.697.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 667).

6.697.3.6 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 668).

6.697.3.7 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h

6.698 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3249).

#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.698.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3249). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.698.2 Constructor & Destructor Documentation

6.698.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.698.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.698.3 Member Function Documentation

6.698.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.698.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.698.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 672).

6.698.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 673).

6.698.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 674).

6.698.3.6 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 675).

6.698.3.7 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 676).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h

6.699 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3253).

#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.699.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3253). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.699.2 Constructor & Destructor Documentation

6.699.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.699.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.699.3 Member Function Documentation

6.699.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.699.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.699.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 658).

6.699.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 659).

6.699.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 660).

6.699.3.6 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 661).

6.699.3.7 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 662).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h

6.700 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3257).

#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.700.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3257). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.700.2 Constructor & Destructor Documentation

6.700.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.700.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.700.3 Member Function Documentation

6.700.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.700.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.700.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 679).

6.700.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 680).

6.700.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 681).

6.700.3.6 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 682).

6.700.3.7 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**

6.701 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3261).

#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.701.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3261). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.701.2 Constructor & Destructor Documentation

6.701.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.701.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.701.3 Member Function Documentation

6.701.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.701.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.701.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 686).

6.701.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 687).

6.701.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 688).

6.701.3.6 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 689).

6.701.3.7 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 690).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h**

6.702 activemq::state::TransactionState Class Reference

```
#include <src/main/activemq/state/TransactionState.h>
```

Public Member Functions

- **TransactionState** (const **Pointer**< **TransactionId** > &id)
- virtual **~TransactionState** ()
- **std::string toString** () const
- void **addCommand** (const **Pointer**< **Command** > &operation)
- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > &**getCommands** () const
- const **Pointer**< **TransactionId** > &**getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const

6.702.1 Constructor & Destructor Documentation

6.702.1.1 `activemq::state::TransactionState::TransactionState (const Pointer< TransactionId > & id)`

6.702.1.2 `virtual activemq::state::TransactionState::~~TransactionState ()`
[virtual]

6.702.2 Member Function Documentation

6.702.2.1 `void activemq::state::TransactionState::addCommand (const Pointer< Command > & operation)`

6.702.2.2 `void activemq::state::TransactionState::checkShutdown () const`

6.702.2.3 `const StlList< Pointer<Command> >& activemq::state::TransactionState::getCommands () const`
[inline]

6.702.2.4 `const Pointer<TransactionId>& activemq::state::TransactionState::getId () const` [inline]

6.702.2.5 `int activemq::state::TransactionState::getPreparedResult () const`
[inline]

6.702.2.6 `bool activemq::state::TransactionState::isPrepared () const` [inline]

6.702.2.7 `void activemq::state::TransactionState::setPrepared (bool prepared)`
[inline]

6.702.2.8 `void activemq::state::TransactionState::setPreparedResult (int preparedResult)` [inline]

6.702.2.9 `void activemq::state::TransactionState::shutdown ()` [inline]

6.702.2.10 `std::string activemq::state::TransactionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/TransactionState.h`

6.703 decaf::internal::util::concurrent::Transferer< E > Class Template Reference

Shared internal API for dual stacks and queues.

`#include <src/main/decaf/internal/util/concurrent/Transferer.h>`Inheritance diagram for decaf::internal::util::concurrent::Transferer< E >:

6.703.1 Detailed Description

`template<typename E> class decaf::internal::util::concurrent::Transferer< E >`

Shared internal API for dual stacks and queues.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/Transferer.h`

6.704 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

#include <src/main/decaf/internal/util/concurrent/TransferQueue.h> Inheritance diagram for decaf::internal::util::concurrent::TransferQueue< E >:

Public Member Functions

- **TransferQueue** ()

*Node class for **TransferQueue** (p. 3268).*

- virtual ~**TransferQueue** ()
- virtual void **transfer** (E *e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)

Performs a put.

- virtual E * **transfer** (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)

Performs a take.

6.704.1 Detailed Description

```
template<typename E> class decaf::internal::util::concurrent::TransferQueue< E >
```

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers. The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

6.704.2 Constructor & Destructor Documentation

6.704.2.1 template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue () [inline]

Node class for **TransferQueue** (p. 3268). Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

6.704.2.2 `template<typename E > virtual
decaf::internal::util::concurrent::TransferQueue< E
>::~~TransferQueue () [inline, virtual]`

6.704.3 Member Function Documentation

6.704.3.1 `template<typename E > virtual E*
decaf::internal::util::concurrent::TransferQueue< E
>::transfer (bool timed, long long nanos) throw
(decaf::util::concurrent::TimeoutException, de-
caf::lang::exceptions::InterruptedException) [inline,
virtual]`

Performs a take.

Parameters:

timed if this operation should timeout
nanos the timeout, in nanoseconds

Returns:

the item provided or received;

Exceptions:

TimeoutException if the operation timed out waiting for the producer to offer an item.
InterruptedException if the thread was interrupted while waiting for the producer to offer an item.

Implements `decaf::internal::util::concurrent::Transferer< E >` (p. 3267).

6.704.3.2 `template<typename E > virtual void
decaf::internal::util::concurrent::TransferQueue< E
>::transfer (E * e, bool timed, long long nanos)
throw (decaf::util::concurrent::TimeoutException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Performs a put.

Parameters:

e the item to be handed to a consumer;
timed if this operation should timeout
nanos the timeout, in nanoseconds

Exceptions:

TimeoutException if the operation timed out waiting for the consumer to accept the item offered.
InterruptedException if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3267).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/TransferQueue.h`

6.705 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

#include <src/main/decaf/internal/util/concurrent/TransferStack.h> Inheritance diagram for decaf::internal::util::concurrent::TransferStack< E >:

Public Member Functions

- **TransferStack** ()
- virtual **~TransferStack** ()
- virtual void **transfer** (E *e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a put.
- virtual E * **transfer** (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a take.

```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

6.705.1 Constructor & Destructor Documentation

- 6.705.1.1** template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack () [inline]
- 6.705.1.2** template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack () [inline, virtual]

6.705.2 Member Function Documentation

- 6.705.2.1** template<typename E > virtual E* decaf::internal::util::concurrent::TransferStack< E >::transfer (bool *timed*, long long *nanos*) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Performs a take.

Parameters:

timed if this operation should timeout
nanos the timeout, in nanoseconds

Returns:

the item provided or received;

Exceptions:

TimeoutException if the operation timed out waiting for the producer to offer an item.

InterruptedException if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3267).

```
6.705.2.2  template<typename E > virtual void
           decaf::internal::util::concurrent::TransferStack< E
           >::transfer (E * e,  bool timed,  long long nanos)
           throw ( decaf::util::concurrent::TimeoutException,
           decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Performs a put.

Parameters:

e the item to be handed to a consumer;

timed if this operation should timeout

nanos the timeout, in nanoseconds

Exceptions:

TimeoutException if the operation timed out waiting for the consumer to accept the item offered.

InterruptedException if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3267).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferStack.h**

6.706 activemq::transport::Transport Class Reference

Interface for a transport layer for command objects.

#include <src/main/activemq/transport/Transport.h> Inheritance diagram for activemq::transport::Transport:

Public Member Functions

- virtual **~Transport** ()
- virtual void **start** ()=0 throw (decaf::io::IOException)
*Starts the **Transport** (p. 3273), the send methods of a **Transport** (p. 3273) will throw an exception if used before the **Transport** (p. 3273) is started.*
- virtual void **stop** ()=0 throw (decaf::io::IOException)
*Stops the **Transport** (p. 3273).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)=0
*Sets the **WireFormat** instance to use.*
- virtual void **setTransportListener** (**TransportListener** *listener)=0
Sets the observer of asynchronous events from this transport.
- virtual **TransportListener** * **getTransportListener** () const =0
Gets the observer of asynchronous events from this transport.
- virtual **Transport** * **narrow** (const std::type_info &typeId)=0
*Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const =0
*Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const =0

Is the **Transport** (p. 3273) Connected to its Broker.

- virtual bool **isClosed** () const =0

Has the **Transport** (p. 3273) been shutdown and no longer usable.

- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0 throw (decaf::io::IOException)
reconnect to another location

6.706.1 Detailed Description

Interface for a transport layer for command objects. Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 3273) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 3273) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

6.706.2 Constructor & Destructor Documentation

6.706.2.1 virtual **activemq::transport::Transport::~~Transport** () [inline, virtual]

6.706.3 Member Function Documentation

6.706.3.1 virtual std::string **activemq::transport::Transport::getRemoteAddress** ()
const [pure virtual]

Returns:

the remote address for this connection

Implemented in **activemq::transport::failover::FailoverTransport** (p.1616), **activemq::transport::IOTransport** (p.1825), **activemq::transport::mock::MockTransport** (p.2374), and **activemq::transport::TransportFilter** (p.3284).

6.706.3.2 virtual **TransportListener*** **activemq::transport::Transport::getTransportListener** ()
const [pure virtual]

Gets the observer of asynchronous events from this transport.

Returns:

the listener of transport events.

Implemented in **activemq::transport::failover::FailoverTransport** (p.1616), **activemq::transport::IOTransport** (p.1825), **activemq::transport::mock::MockTransport** (p.2375), and **activemq::transport::TransportFilter** (p.3284).

6.706.3.3 virtual bool activemq::transport::Transport::isClosed () const [pure virtual]

Has the **Transport** (p. 3273) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 3273)

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1617), **activemq::transport::IOTransport** (p. 1825), **activemq::transport::mock::MockTransport** (p. 2375), **activemq::transport::tcp::TcpTransport** (p. 3161), and **activemq::transport::TransportFilter** (p. 3284).

6.706.3.4 virtual bool activemq::transport::Transport::isConnected () const [pure virtual]

Is the **Transport** (p. 3273) Connected to its Broker.

Returns:

true if a connection has been made.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1617), **activemq::transport::IOTransport** (p. 1826), **activemq::transport::mock::MockTransport** (p. 2375), **activemq::transport::tcp::TcpTransport** (p. 3161), and **activemq::transport::TransportFilter** (p. 3284).

6.706.3.5 virtual bool activemq::transport::Transport::isFaultTolerant () const [pure virtual]

Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 3273) is fault tolerant.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1617), **activemq::transport::IOTransport** (p. 1826), **activemq::transport::mock::MockTransport** (p. 2376), **activemq::transport::tcp::TcpTransport** (p. 3162), and **activemq::transport::TransportFilter** (p. 3284).

6.706.3.6 virtual Transport* activemq::transport::Transport::narrow (const std::type_info & typeId) [pure virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1618), `activemq::transport::IOTransport` (p. 1826), `activemq::transport::mock::MockTransport` (p. 2376), and `activemq::transport::TransportFilter` (p. 3285).

Referenced by `activemq::transport::failover::FailoverTransport::narrow()`.

6.706.3.7 `virtual void activemq::transport::Transport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2788), `activemq::transport::failover::FailoverTransport` (p. 1619), `activemq::transport::inactivity::InactivityMonitor` (p. 1735), `activemq::transport::IOTransport` (p. 1826), `activemq::transport::logging::LoggingTransport` (p. 2043), `activemq::transport::mock::MockTransport` (p. 2376), `activemq::transport::TransportFilter` (p. 3285), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2463).

6.706.3.8 `virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & uri) throw (decaf::io::IOException) [pure virtual]`

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1619), and `activemq::transport::TransportFilter` (p. 3286).

6.706.3.9 `virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sends the given command to the broker and then waits for the response.

Parameters:

command - The command to be sent.
timeout - The time to wait for this response.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.
UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2789), **activemq::transport::failover::FailoverTransport** (p. 1620), **activemq::transport::IOTransport** (p. 1827), **activemq::transport::logging::LoggingTransport** (p. 2043), **activemq::transport::mock::MockTransport** (p. 2377), **activemq::transport::TransportFilter** (p. 3286), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2464).

```
6.706.3.10 virtual Pointer<Response> activemq::transport::Transport::request
            (const Pointer< Command > & command)
            throw ( decaf::io::IOException, de-
                    caf::lang::exceptions::UnsupportedOperationException )
            [pure virtual]
```

Sends the given command to the broker and then waits for the response.

Parameters:

command the command to be sent.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.
UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2789), **activemq::transport::failover::FailoverTransport** (p. 1620), **activemq::transport::IOTransport** (p. 1827), **activemq::transport::logging::LoggingTransport** (p. 2044), **activemq::transport::mock::MockTransport** (p. 2377), **activemq::transport::TransportFilter** (p. 3287), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2464).

```
6.706.3.11 virtual void activemq::transport::Transport::setTransportListener
            (TransportListener * listener) [pure virtual]
```

Sets the observer of asynchronous events from this transport.

Parameters:

listener the listener of transport events.

Implemented in `activemq::transport::failover::FailoverTransport` (p.1622), `activemq::transport::IOTransport` (p.1828), `activemq::transport::mock::MockTransport` (p.2380), and `activemq::transport::TransportFilter` (p.3287).

6.706.3.12 `virtual void activemq::transport::Transport::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [pure virtual]`

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode commands.

Implemented in `activemq::transport::IOTransport` (p.1828), and `activemq::transport::TransportFilter` (p.3287).

6.706.3.13 `virtual void activemq::transport::Transport::start () throw (decaf::io::IOException) [pure virtual]`

Starts the **Transport** (p.3273), the send methods of a **Transport** (p.3273) will throw an exception if used before the **Transport** (p.3273) is started.

Exceptions:

IOException if and error occurs while starting the **Transport** (p.3273).

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p.2790), `activemq::transport::failover::FailoverTransport` (p.1623), `activemq::transport::IOTransport` (p.1828), `activemq::transport::mock::MockTransport` (p.2380), `activemq::transport::TransportFilter` (p.3287), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p.2465).

6.706.3.14 `virtual void activemq::transport::Transport::stop () throw (decaf::io::IOException) [pure virtual]`

Stops the **Transport** (p.3273).

Exceptions:

IOException if an error occurs while stopping the transport.

Implemented in `activemq::transport::failover::FailoverTransport` (p.1623), `activemq::transport::IOTransport` (p.1829), `activemq::transport::mock::MockTransport` (p.2380), and `activemq::transport::TransportFilter` (p.3288).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/Transport.h`

6.707 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

#include <src/main/activemq/transport/TransportFactory.h> Inheritance diagram for activemq::transport::TransportFactory:

Public Member Functions

- virtual **~TransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location)=0 throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3273) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)=0 throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3273) instance which can be used in composite transport instances.*

6.707.1 Detailed Description

Defines the interface for Factories that create Transports or TransportFilters. The factory should be able to create either a completely configured **Transport** (p. 3273) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimed down version that is used in composite transports like Failover or Fanout.

Since:

3.0

6.707.2 Constructor & Destructor Documentation

- 6.707.2.1** virtual **activemq::transport::TransportFactory::~~TransportFactory** ()
 [inline, virtual]

6.707.3 Member Function Documentation

- 6.707.3.1** virtual **Pointer<Transport> activemq::transport::TransportFactory::create** (const **decaf::net::URI** & *location*) throw (exceptions::ActiveMQException)
 [pure virtual]

Creates a fully configured **Transport** (p. 3273) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implemented in `activemq::transport::failover::FailoverTransportFactory` (p. 1625), `activemq::transport::mock::MockTransportFactory` (p. 2383), and `activemq::transport::tcp::TcpTransportFactory` (p. 3164).

6.707.3.2 `virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite (const decaf::net::URI & location) throw (exceptions::ActiveMQException)`
[pure virtual]

Creates a slimmed down **Transport** (p. 3273) instance which can be used in composite transport instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQException if an error occurs

Implemented in `activemq::transport::failover::FailoverTransportFactory` (p. 1625), `activemq::transport::mock::MockTransportFactory` (p. 2383), and `activemq::transport::tcp::TcpTransportFactory` (p. 3164).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFactory.h`

6.708 activemq::transport::TransportFilter Class Reference

A filter on the transport layer.

#include <src/main/activemq/transport/TransportFilter.h> Inheritance diagram for activemq::transport::TransportFilter:

Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual **~TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Not supported by this class - throws an exception.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **start** () throw (**decaf::io::IOException**)

Starts this transport object and creates the thread for polling on the input stream for commands.

- virtual void **stop** () throw (decaf::io::IOException)

*Stops the **Transport** (p. 3273).*

- virtual void **close** () throw (decaf::io::IOException)

Stops the polling thread and closes the streams.

- virtual **Transport** * **narrow** (const std::type_info &typeId)

*Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3273) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3273) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const

- virtual void **reconnect** (const decaf::net::URI &uri) throw (decaf::io::IOException)

reconnect to another location

Protected Member Functions

- void **fire** (const decaf::lang::Exception &ex)

Notify the listener of the thrown Exception.

- void **fire** (const Pointer< Command > &command)

Notify the listener of the new incoming Command.

Protected Attributes

- Pointer< Transport > **next**

The transport that this filter wraps around.

- TransportListener * **listener**

Listener of this transport.

6.708.1 Detailed Description

A filter on the transport layer. **Transport** (p. 3273) filters implement the **Transport** (p. 3273) interface and optionally delegate calls to another **Transport** (p. 3273) object.

Since:

1.0

6.708.2 Constructor & Destructor Documentation

6.708.2.1 activemq::transport::TransportFilter::TransportFilter (const Pointer< Transport > & *next*)

Constructor.

Parameters:

next - the next **Transport** (p. 3273) in the chain

6.708.2.2 virtual activemq::transport::TransportFilter::~~TransportFilter () [inline, virtual]

6.708.3 Member Function Documentation

6.708.3.1 virtual void activemq::transport::TransportFilter::close () throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

IOException if an error occurs while closing the **Transport** (p. 3273).

Implements **decaf::io::Closeable** (p. 1019).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2788), **activemq::transport::inactivity::InactivityMonitor** (p. 1734), **activemq::transport::tcp::TcpTransport** (p. 3161), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2463).

6.708.3.2 void activemq::transport::TransportFilter::fire (const Pointer< Command > & *command*) [protected]

Notify the listener of the new incoming Command.

Parameters:

command - the command to send to the listener

6.708.3.3 `void activemq::transport::TransportFilter::fire (const decaf::lang::Exception & ex)` [protected]

Notify the listener of the thrown Exception.

Parameters:

ex - the exception to send to listeners

6.708.3.4 `virtual std::string activemq::transport::TransportFilter::getRemoteAddress () const` [inline, virtual]

Returns:

the remote address for this connection

Implements `activemq::transport::Transport` (p. 3274).

6.708.3.5 `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener () const` [inline, virtual]

Gets the observer of asynchronous exceptions from this transport.

Returns:

The listener of transport events.

Implements `activemq::transport::Transport` (p. 3274).

6.708.3.6 `virtual bool activemq::transport::TransportFilter::isClosed () const` [inline, virtual]

Has the `Transport` (p. 3273) been shutdown and no longer usable.

Returns:

true if the `Transport` (p. 3273)

Implements `activemq::transport::Transport` (p. 3275).

Reimplemented in `activemq::transport::tcp::TcpTransport` (p. 3161).

6.708.3.7 `virtual bool activemq::transport::TransportFilter::isConnected () const` [inline, virtual]

Is the `Transport` (p. 3273) Connected to its Broker.

Returns:

true if a connection has been made.

Implements `activemq::transport::Transport` (p. 3275).

Reimplemented in `activemq::transport::tcp::TcpTransport` (p. 3161).

6.708.3.8 virtual bool activemq::transport::TransportFilter::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 3273) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 3273) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3275).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3162).

6.708.3.9 virtual Transport* activemq::transport::TransportFilter::narrow (const std::type_info & typeId) [virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3273) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3275).

6.708.3.10 virtual void activemq::transport::TransportFilter::onCommand (const Pointer< Command > & command) [virtual]

Event handler for the receipt of a command.

Parameters:

command - the received command object.

Implements **activemq::transport::TransportListener** (p. 3289).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2788), **activemq::transport::inactivity::InactivityMonitor** (p. 1734), **activemq::transport::logging::LoggingTransport** (p. 2043), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2463).

6.708.3.11 virtual void activemq::transport::TransportFilter::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3276).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2788), **activemq::transport::inactivity::InactivityMonitor** (p. 1735), **activemq::transport::logging::LoggingTransport** (p. 2043), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2463).

6.708.3.12 **virtual void activemq::transport::TransportFilter::onException (const decaf::lang::Exception & *ex*)** [virtual]

Event handler for an exception from a command transport.

Parameters:

ex The exception to handle.

Implements **activemq::transport::TransportListener** (p. 3290).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 1735).

6.708.3.13 **virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & *uri*) throw (decaf::io::IOException)** [virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

Implements **activemq::transport::Transport** (p. 3276).

6.708.3.14 **virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)** [inline, virtual]

Not supported by this class - throws an exception.

Parameters:

command - The command that is sent as a request

timeout - The the time to wait for a response.

Exceptions:

IOException

UnsupportedOperationException.

Implements `activemq::transport::Transport` (p. 3276).

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2789), `activemq::transport::logging::LoggingTransport` (p. 2043), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2464).

6.708.3.15 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Not supported by this class - throws an exception.

Parameters:

command the command that is sent as a request

Exceptions:

IOException

UnsupportedOperationException.

Implements `activemq::transport::Transport` (p. 3277).

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2789), `activemq::transport::logging::LoggingTransport` (p. 2044), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2464).

6.708.3.16 `virtual void activemq::transport::TransportFilter::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters:

listener the listener of transport events.

Implements `activemq::transport::Transport` (p. 3277).

6.708.3.17 `virtual void activemq::transport::TransportFilter::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode commands.

Implements `activemq::transport::Transport` (p. 3278).

6.708.3.18 `virtual void activemq::transport::TransportFilter::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands. If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

IOException if an error occurs or if this transport has already been closed.

Implements `activemq::transport::Transport` (p. 3278).

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2790), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2465).

6.708.3.19 `virtual void activemq::transport::TransportFilter::stop () throw (decaf::io::IOException) [virtual]`

Stops the `Transport` (p. 3273).

Exceptions:

IOException if an error occurs while stopping the `Transport` (p. 3273).

Implements `activemq::transport::Transport` (p. 3278).

6.708.3.20 `virtual void activemq::transport::TransportFilter::transportInterrupted () [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3290).

6.708.3.21 `virtual void activemq::transport::TransportFilter::transportResumed () [virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3290).

6.708.4 Field Documentation

6.708.4.1 `TransportListener* activemq::transport::TransportFilter::listener [protected]`

Listener of this transport.

6.708.4.2 `Pointer<Transport> activemq::transport::TransportFilter::next [protected]`

The transport that this filter wraps around.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFilter.h`

6.709 activemq::transport::TransportListener Class Reference

A listener of asynchronous exceptions from a command transport object.

#include <src/main/activemq/transport/TransportListener.h> Inheritance diagram for activemq::transport::TransportListener:

Public Member Functions

- virtual **~TransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)=0
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)=0
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()=0
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()=0
The transport has resumed after an interruption.

6.709.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

6.709.2 Constructor & Destructor Documentation

- 6.709.2.1** virtual **activemq::transport::TransportListener::~TransportListener** ()
[inline, virtual]

6.709.3 Member Function Documentation

- 6.709.3.1** virtual void **activemq::transport::TransportListener::onCommand** (const **Pointer**< **Command** > & *command*) [pure virtual]

Event handler for the receipt of a command. The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3273) deletes the command upon receipt.

Parameters:

command the received command object.

Implemented in **activemq::core::ActiveMQConnection** (p. 240), **ac-**
tivemq::transport::correlator::ResponseCorrelator (p. 2788), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1628), **ac-**
tivemq::transport::inactivity::InactivityMonitor (p. 1734), **ac-**
tivemq::transport::logging::LoggingTransport (p. 2043), **ac-**
tivemq::transport::mock::InternalCommandListener (p. 1800),
activemq::transport::TransportFilter (p. 3285), and **ac-**
tivemq::wireformat::openwire::OpenWireFormatNegotiator (p. 2463).

6.709.3.2 virtual void activemq::transport::TransportListener::onException (const decaf::lang::Exception & *ex*) [pure virtual]

Event handler for an exception from a command transport.

Parameters:

ex The exception being propagated to this listener to handle.

Implemented in **activemq::core::ActiveMQConnection** (p. 240),
activemq::transport::failover::BackupTransport (p. 645), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1628), **ac-**
tivemq::transport::inactivity::InactivityMonitor (p. 1735), and **ac-**
tivemq::transport::TransportFilter (p. 3286).

6.709.3.3 virtual void activemq::transport::TransportListener::transportInterrupted () [pure virtual]

The transport has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 242),
activemq::transport::DefaultTransportListener (p. 1476), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1628), and **ac-**
tivemq::transport::TransportFilter (p. 3288).

6.709.3.4 virtual void activemq::transport::TransportListener::transportResumed () [pure virtual]

The transport has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 242),
activemq::transport::DefaultTransportListener (p. 1476), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1628), and **ac-**
tivemq::transport::TransportFilter (p. 3288).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportListener.h**

6.710 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 3273) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```

Public Member Functions

- virtual **~TransportRegistry** ()
- **TransportFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)

*Gets a Registered **TransportFactory** (p. 3279) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **TransportFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

*Registers a new **TransportFactory** (p. 3279) with this Registry.*

- void **unregisterFactory** (const std::string &name)

Unregisters the Factory with the given name and deletes that instance of the Factory.

- std::vector< std::string > **getTransportNames** () const

Retrieves a list of the names of all the Registered Transport's in this Registry.

Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

*Gets the single instance of the **TransportRegistry** (p. 3291).*

6.710.1 Detailed Description

Registry of all **Transport** (p. 3273) Factories that are available to the client at runtime. New Transport's must have a factory registered here before a connection attempt is made.

Since:

3.0

6.710.2 Constructor & Destructor Documentation

6.710.2.1 `virtual activemq::transport::TransportRegistry::~TransportRegistry ()`
[virtual]

6.710.3 Member Function Documentation

6.710.3.1 `TransportFactory* activemq::transport::TransportRegistry::findFactory (const std::string & name) const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets a Registered **TransportFactory** (p. 3279) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters:

name The name of the Factory to find in the Registry.

Returns:

the Factory registered under the given name.

Exceptions:

NoSuchElementException if no factory is registered with that name.

6.710.3.2 `static TransportRegistry& activemq::transport::TransportRegistry::getInstance ()`
[static]

Gets the single instance of the **TransportRegistry** (p. 3291).

Returns:

reference to the single instance of this Registry

6.710.3.3 `std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames () const`

Retrieves a list of the names of all the Registered Transport's in this Registry.

Returns:

std vector of strings with all the **Transport** (p. 3273) names registered.

6.710.3.4 `void activemq::transport::TransportRegistry::registerFactory (const std::string & name, TransportFactory * factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)`

Registers a new **TransportFactory** (p. 3279) with this Registry. If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters:

- name* The name of the new Factory to register.
factory The new Factory to add to the Registry.

Exceptions:

- IllegalArgumentException* if name is the empty string.
NullPointerException if the Factory is Null.

6.710.3.5 void activemq::transport::TransportRegistry::unregisterFactory (const std::string & name)

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters:

- name* Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

6.711 decaf::net::UnknownHostException Class Reference

#include <src/main/decaf/net/UnknownHostException.h> Inheritance diagram for decaf::net::UnknownHostException:

Public Member Functions

- **UnknownHostException** () throw ()
Default Constructor.
- **UnknownHostException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()
Copy Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownHostException** (const std::exception *cause) throw ()
Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownHostException** * clone () const
Clones this exception.
- virtual ~**UnknownHostException** () throw ()

6.711.1 Constructor & Destructor Documentation

6.711.1.1 decaf::net::UnknownHostException::UnknownHostException () throw () [inline]

Default Constructor.

6.711.1.2 decaf::net::UnknownHostException::UnknownHostException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.711.1.3 `decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.711.1.4 `decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.711.1.5 `decaf::net::UnknownHostException::UnknownHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.711.1.6 `decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.711.1.7 virtual decaf::net::UnknownHostException::~~UnknownHostException ()
throw () [inline, virtual]

6.711.2 Member Function Documentation

6.711.2.1 virtual UnknownHostException* decaf::net::UnknownHostException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownHostException.h**

6.712 decaf::net::UnknownServiceException Class Reference

#include <src/main/decaf/net/UnknownServiceException.h> Inheritance diagram for decaf::net::UnknownServiceException:

Public Member Functions

- **UnknownServiceException** () throw ()
Default Constructor.
- **UnknownServiceException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()
Copy Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownServiceException** (const std::exception *cause) throw ()
Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownServiceException** * clone () const
Clones this exception.
- virtual ~**UnknownServiceException** () throw ()

6.712.1 Constructor & Destructor Documentation

6.712.1.1 decaf::net::UnknownServiceException::UnknownServiceException () throw () [inline]

Default Constructor.

6.712.1.2 decaf::net::UnknownServiceException::UnknownServiceException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.712.1.3 decaf::net::UnknownServiceException::UnknownServiceException (const UnknownServiceException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.712.1.4 decaf::net::UnknownServiceException::UnknownServiceException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.712.1.5 decaf::net::UnknownServiceException::UnknownServiceException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.712.1.6 decaf::net::UnknownServiceException::UnknownServiceException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.712.1.7 **virtual**
decaf::net::UnknownServiceException::~UnknownServiceException ()
throw () [inline, virtual]

6.712.2 Member Function Documentation

6.712.2.1 **virtual UnknownServiceException* de-**
caf::net::UnknownServiceException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownServiceException.h`

6.713 decaf::io::UnsupportedEncodingException Class Reference

Thrown when the the Character Encoding is not supported.

#include <src/main/decaf/io/UnsupportedEncodingException.h> Inheritance diagram for decaf::io::UnsupportedEncodingException:

Public Member Functions

- **UnsupportedEncodingException** () throw ()
Default Constructor.
- **UnsupportedEncodingException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **UnsupportedEncodingException** (const **UnsupportedEncodingException** &ex) throw ()
Copy Constructor.
- **UnsupportedEncodingException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedEncodingException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UnsupportedEncodingException** * clone () const
Clones this exception.
- virtual ~**UnsupportedEncodingException** () throw ()

6.713.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since:

1.0

6.713.2 Constructor & Destructor Documentation

6.713.2.1 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException () throw () [inline]

Default Constructor.

6.713.2.2 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex the exception to copy

6.713.2.3 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const UnsupportedEncodingException & ex) throw () [inline]`

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.713.2.4 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.713.2.5 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.713.2.6 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.713.2.7 **virtual**
decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException
() throw () [inline, virtual]

6.713.3 Member Function Documentation

6.713.3.1 **virtual** **UnsupportedEncodingException*** **decaf::io::UnsupportedEncodingException::clone () const** [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UnsupportedEncodingException.h`

6.714 decaf::lang::exceptions::UnsupportedOperationException Class Reference

#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h> Inheritance diagram for decaf::lang::exceptions::UnsupportedOperationException:

Public Member Functions

- **UnsupportedOperationException** () throw ()
Default Constructor.
- **UnsupportedOperationException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1574).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
Copy Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedOperationException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnsupportedOperationException** * clone () const
Clones this exception.
- virtual ~**UnsupportedOperationException** () throw ()

6.714.1 Constructor & Destructor Documentation

6.714.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException () throw () [inline]

Default Constructor.

6.714.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1574).

Parameters:

ex An exception that should become this type of **Exception** (p. 1574)

6.714.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const UnsupportedOperationException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of **Exception** (p. 1574)

6.714.1.4 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const char * *file*, const int *lineNumber*, const std::exception * *cause*,
const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.714.1.5 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2497) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.714.1.6 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()
[inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.714.1.7 **virtual**
decaf::lang::exceptions::UnsupportedOperationException::~~UnsupportedOperationException
() throw () [inline, virtual]

6.714.2 Member Function Documentation

6.714.2.1 **virtual UnsupportedOperationException* de-**
caf::lang::exceptions::UnsupportedOperationException::clone () const
[inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1574) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1577).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 2687).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/UnsupportedOperationException.h`

6.715 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

#include <src/main/cms/UnsupportedOperationException.h> Inheritance diagram for cms::UnsupportedOperationException:

Public Member Functions

- **UnsupportedOperationException** () throw ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**UnsupportedOperationException** () throw ()

6.715.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Since:

2.0

6.715.2 Constructor & Destructor Documentation

- 6.715.2.1 **cms::UnsupportedOperationException::UnsupportedOperationException** () throw ()
- 6.715.2.2 **cms::UnsupportedOperationException::UnsupportedOperationException** (const **UnsupportedOperationException** & *ex*) throw ()
- 6.715.2.3 **cms::UnsupportedOperationException::UnsupportedOperationException** (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.715.2.4 **cms::UnsupportedOperationException::UnsupportedOperationException** (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.715.2.5 virtual **cms::UnsupportedOperationException::~~UnsupportedOperationException** () throw () [virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/UnsupportedOperationException.h`

6.716 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 3308) as defined by RFC 2396.

#include <src/main/decaf/net/URI.h> Inheritance diagram for decaf::net::URI:

Public Member Functions

- **URI** ()
Default Constructor, same as calling a Constructor with all fields empty.
- **URI** (const **URI** &uri) throw (URISyntaxException)
*Constructs a **URI** (p. 3308) as a copy of another **URI** (p. 3308).*
- **URI** (const std::string &uri) throw (URISyntaxException)
*Constructs a **URI** (p. 3308) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3308) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3308) from the given components.*
- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3308) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3308) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **URI** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const

- `std::string getPath () const`
- `int getPort () const`
- `std::string getQuery () const`
- `std::string getScheme () const`
- `std::string getUserInfo () const`
- `std::string getRawAuthority () const`
*Returns the raw authority component of this **URI** (p. 3308).*
- `std::string getRawFragment () const`
*Returns the raw fragment component of this **URI** (p. 3308).*
- `std::string getRawPath () const`
*Returns the raw path component of this **URI** (p. 3308).*
- `std::string getRawQuery () const`
*Returns the raw query component of this **URI** (p. 3308).*
- `std::string getRawSchemeSpecificPart () const`
*Returns the raw scheme-specific part of this **URI** (p. 3308).*
- `std::string getSchemeSpecificPart () const`
*Returns the decoded scheme-specific part of this **URI** (p. 3308).*
- `std::string getRawUserInfo () const`
*Returns the raw user-information component of this **URI** (p. 3308).*
- `bool isAbsolute () const`
*Tells whether or not this **URI** (p. 3308) is absolute.*
- `bool isOpaque () const`
*Tells whether or not this **URI** (p. 3308) is opaque.*
- `URI normalize () const`
*Normalizes this **URI**'s path.*
- `URI parseServerAuthority () const throw (URISyntaxException)`
*Attempts to parse this **URI**'s authority component, if defined, into user-information, host, and port components.*
- `URI relativize (const URI &uri) const`
*Relativizes the given **URI** (p. 3308) against this **URI** (p. 3308).*
- `URI resolve (const std::string &str) const throw (lang::exceptions::IllegalArgumentException)`
*Constructs a new **URI** (p. 3308) by parsing the given string and then resolving it against this **URI** (p. 3308).*
- `URI resolve (const URI &uri) const`
*Resolves the given **URI** (p. 3308) against this **URI** (p. 3308).*

- `std::string toString () const`
*Returns the content of this **URI** (p. 3308) as a string.*
- `URL toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)`
*Constructs a **URL** (p. 3347) from this **URI** (p. 3308).*

Static Public Member Functions

- `static URI create (const std::string uri) throw (lang::exceptions::IllegalArgumentException)`
*Creates a **URI** (p. 3308) by parsing the given string.*

6.716.1 Detailed Description

This class represents an instance of a **URI** (p. 3308) as defined by RFC 2396.

6.716.2 Constructor & Destructor Documentation

6.716.2.1 decaf::net::URI::URI ()

Default Constructor, same as calling a Constructor with all fields empty.

6.716.2.2 decaf::net::URI::URI (const URI & uri) throw (URISyntaxException)

Constructs a **URI** (p. 3308) as a copy of another **URI** (p. 3308).

Parameters:

uri - uri to copy

6.716.2.3 decaf::net::URI::URI (const std::string & uri) throw (URISyntaxException)

Constructs a **URI** (p. 3308) from the given string.

Parameters:

uri - string uri to parse.

6.716.2.4 decaf::net::URI::URI (const std::string & scheme, const std::string & ssp, const std::string & fragment) throw (URISyntaxException)

Constructs a **URI** (p. 3308) from the given components.

Parameters:

scheme - the uri scheme

ssp - Scheme specific part

fragment - Fragment

6.716.2.5 `decaf::net::URI::URI (const std::string & scheme, const std::string & userInfo, const std::string & host, int port, const std::string & path, const std::string & query, const std::string & fragment) throw (URISyntaxException)`

Constructs a **URI** (p. 3308) from the given components.

Parameters:

scheme - Scheme name

userInfo - User name and authorization information

host - Host name

port - Port number

path - Path

query - Query

fragment - Fragment

6.716.2.6 `decaf::net::URI::URI (const std::string & scheme, const std::string & host, const std::string & path, const std::string & fragment) throw (URISyntaxException)`

Constructs a **URI** (p. 3308) from the given components.

Parameters:

scheme - Scheme name

host - Host name

path - Path

fragment - Fragment

6.716.2.7 `decaf::net::URI::URI (const std::string & scheme, const std::string & authority, const std::string & path, const std::string & query, const std::string & fragment) throw (URISyntaxException)`

Constructs a **URI** (p. 3308) from the given components.

Parameters:

scheme - Scheme name

authority - Authority

path - Path

query - Query

fragment - Fragment

6.716.2.8 virtual decaf::net::URI::~~URI () [inline, virtual]

6.716.3 Member Function Documentation

6.716.3.1 virtual int decaf::net::URI::compareTo (const URI & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the value to compare to this one.

Returns:

zero if equal minus one if less than and one if greater than.

6.716.3.2 static URI decaf::net::URI::create (const std::string *uri*) throw (lang::exceptions::IllegalArgumentException) [static]

Creates a **URI** (p. 3308) by parsing the given string. This convenience factory method works as if by invoking the URI(string) constructor; any **URISyntaxException** (p. 3335) thrown by the constructor is caught and wrapped in a new IllegalArgumentException object, which is then thrown.

Parameters:

uri - **URI** (p. 3308) string to parse

Exceptions:

IllegalArgumentException

6.716.3.3 virtual bool decaf::net::URI::equals (const URI & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.716.3.4 std::string decaf::net::URI::getAuthority () const

Returns:

the decoded authority component of this **URI** (p. 3308).

6.716.3.5 std::string decaf::net::URI::getFragment () const

Returns:

the decoded fragment component of this **URI** (p. 3308).

6.716.3.6 std::string decaf::net::URI::getHost () const**Returns:**

the host component of this **URI** (p. 3308).

6.716.3.7 std::string decaf::net::URI::getPath () const**Returns:**

the path component of this **URI** (p. 3308).

6.716.3.8 int decaf::net::URI::getPort () const**Returns:**

the port component of this **URI** (p. 3308).

6.716.3.9 std::string decaf::net::URI::getQuery () const**Returns:**

the query component of this **URI** (p. 3308).

6.716.3.10 std::string decaf::net::URI::getRawAuthority () const

Returns the raw authority component of this **URI** (p. 3308). The authority component of a **URI** (p. 3308), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

Returns:

the raw authority component of the **URI** (p. 3308)

6.716.3.11 std::string decaf::net::URI::getRawFragment () const

Returns the raw fragment component of this **URI** (p. 3308). The fragment component of a **URI** (p. 3308), if defined, only contains legal **URI** (p. 3308) characters.

Returns:

the raw fragment component of this **URI** (p. 3308)

6.716.3.12 std::string decaf::net::URI::getRawPath () const

Returns the raw path component of this **URI** (p. 3308). The path component of a **URI** (p. 3308), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

Returns:

the raw path component of this **URI** (p. 3308)

6.716.3.13 std::string decaf::net::URI::getRawQuery () const

Returns the raw query component of this **URI** (p. 3308). The query component of a **URI** (p. 3308), if defined, only contains legal **URI** (p. 3308) characters.

Returns:

the raw query component of the **URI** (p. 3308).

6.716.3.14 std::string decaf::net::URI::getRawSchemeSpecificPart () const

Returns the raw scheme-specific part of this **URI** (p. 3308). The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 3308) only contains legal **URI** (p. 3308) characters.

Returns:

the raw scheme special part of the uri

6.716.3.15 std::string decaf::net::URI::getRawUserInfo () const

Returns the raw user-information component of this **URI** (p. 3308). The user-information component of a **URI** (p. 3308), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

Returns:

the raw user-information component of the **URI** (p. 3308)

6.716.3.16 std::string decaf::net::URI::getScheme () const**Returns:**

the scheme component of this **URI** (p. 3308)

6.716.3.17 std::string decaf::net::URI::getSchemeSpecificPart () const

Returns the decoded scheme-specific part of this **URI** (p. 3308). The string returned by this method is equal to that returned by the getRawSchemeSpecificPart method except that all sequences of escaped octets are decoded.

Returns:

the raw scheme specific part of the uri.

6.716.3.18 `std::string decaf::net::URI::getUserInfo () const`**Returns:**

the user info component of this **URI** (p. 3308)

6.716.3.19 `bool decaf::net::URI::isAbsolute () const`

Tells whether or not this **URI** (p. 3308) is absolute. A **URI** (p. 3308) is absolute if, and only if, it has a scheme component.

Returns:

true if, and only if, this **URI** (p. 3308) is absolute

6.716.3.20 `bool decaf::net::URI::isOpaque () const`

Tells whether or not this **URI** (p. 3308) is opaque. A **URI** (p. 3308) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 3308) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

Returns:

true if, and only if, this **URI** (p. 3308) is opaque

6.716.3.21 `URI decaf::net::URI::normalize () const`

Normalizes this **URI**'s path. If this **URI** (p. 3308) is opaque, or if its path is already in normal form, then this **URI** (p. 3308) is returned. Otherwise a new **URI** (p. 3308) is constructed that is identical to this **URI** (p. 3308) except that its path is computed by normalizing this **URI**'s path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed. 2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable. 3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 3308) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 3308) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more "." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

Returns:

A **URI** (p. 3308) equivalent to this **URI** (p. 3308), but whose path is in normal form

6.716.3.22 `virtual bool decaf::net::URI::operator< (const URI & value) const`
`[virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.716.3.23 `virtual bool decaf::net::URI::operator==(const URI & value) const`
[virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.716.3.24 `URI decaf::net::URI::parseServerAuthority () const throw (`
`URISyntaxException)`

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components. If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 3308) has no authority component, this method simply returns this **URI** (p. 3308).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

Returns:

A **URI** (p. 3308) whose authority field has been parsed as a server-based authority

Exceptions:

URISyntaxException (p. 3335) - If the authority component of this **URI** (p. 3308) is defined but cannot be parsed as a server-based authority.

6.716.3.25 `URI decaf::net::URI::relativize (const URI & uri) const`

Relativizes the given **URI** (p. 3308) against this **URI** (p. 3308). The relativization of the given **URI** (p. 3308) against this **URI** (p. 3308) is computed as follows:

1. If either this **URI** (p. 3308) or the given **URI** (p. 3308) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 3308) is not a prefix of the path of the given **URI** (p. 3308), then the given **URI** (p. 3308) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 3308) is constructed with query and fragment components taken from the given **URI** (p. 3308) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

Parameters:

uri - The **URI** (p. 3308) to be relativized against this **URI** (p. 3308)

Returns:

The resulting **URI** (p. 3308)

6.716.3.26 URI decaf::net::URI::resolve (const URI & uri) const

Resolves the given **URI** (p. 3308) against this **URI** (p. 3308). If the given **URI** (p. 3308) is already absolute, or if this **URI** (p. 3308) is opaque, then a copy of the given **URI** (p. 3308) is returned.

If the given **URI**'s fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 3308) with the given fragment but with all other components equal to those of this **URI** (p. 3308) is returned. This allows a **URI** (p. 3308) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 3308).

Otherwise this method constructs a new hierarchical **URI** (p. 3308) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 3308) is constructed with this **URI**'s scheme and the given **URI**'s query and fragment components. 2. If the given **URI** (p. 3308) has an authority component then the new **URI**'s authority and path are taken from the given **URI** (p. 3308). 3. Otherwise the new **URI**'s authority component is copied from this **URI** (p. 3308), and its path is computed as follows:

1. If the given **URI**'s path is absolute then the new **URI**'s path is taken from the given **URI** (p. 3308). 2. Otherwise the given **URI**'s path is relative, and so the new **URI**'s path is computed by resolving the path of the given **URI** (p. 3308) against the path of this **URI** (p. 3308). This is done by concatenating all but the last segment of this **URI**'s path, if any, with the given **URI**'s path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 3308) is absolute or the given **URI** (p. 3308) is absolute.

Parameters:

uri - The **URI** (p. 3308) to be resolved against this **URI** (p. 3308)

Returns:

The resulting **URI** (p. 3308)

6.716.3.27 URI decaf::net::URI::resolve (const std::string & str) const throw (lang::exceptions::IllegalArgumentException)

Constructs a new **URI** (p. 3308) by parsing the given string and then resolving it against this **URI** (p. 3308). This convenience method works as if invoking it were equivalent to evaluating the expression `resolve(URI::create(str))`.

Parameters:

str - The string to be parsed into a **URI** (p. 3308)

Returns:

The resulting **URI** (p. 3308)

Exceptions:

IllegalArgumentException - If the given string violates RFC 2396

6.716.3.28 std::string decaf::net::URI::toString () const

Returns the content of this **URI** (p. 3308) as a string. If this **URI** (p. 3308) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 3308) was created by normalization, resolution, or relativization, and so a string is constructed from this **URI**'s components according to the rules specified in RFC 2396, section 5.2, step 7.

Returns:

the string form of this **URI** (p. 3308)

6.716.3.29 URL decaf::net::URI::toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)

Constructs a **URL** (p. 3347) from this **URI** (p. 3308). This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 3347)(this.toString())` after first checking that this **URI** (p. 3308) is absolute.

Returns:

A **URL** (p. 3347) constructed from this **URI** (p. 3308)

Exceptions:

IllegalArgumentException - If this **URL** (p. 3347) is not absolute

MalformedURLException (p. 2091) - If a protocol handler for the **URL** (p. 3347) could not be found, or if some other error occurred while constructing the **URL** (p. 3347)

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URI.h**

6.717 decaf::internal::net::URIEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URIEncoderDecoder.h>
```

Public Member Functions

- **URIEncoderDecoder** ()
- virtual **~URIEncoderDecoder** ()

Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)
Validate a string by checking if it contains any characters other than:.
- static void **validateSimple** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)
Validate a string by checking if it contains any characters other than:.
- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)
All characters except letters ('a').
- static std::string **encodeOthers** (const std::string &s)
Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.
- static std::string **decode** (const std::string &s)
Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

6.717.1 Constructor & Destructor Documentation

6.717.1.1 decaf::internal::net::URIEncoderDecoder::URIEncoderDecoder ()

6.717.1.2 virtual decaf::internal::net::URIEncoderDecoder::~~URIEncoderDecoder () [inline, virtual]

6.717.2 Member Function Documentation

6.717.2.1 static std::string decaf::internal::net::URIEncoderDecoder::decode (const std::string & s) [static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme. " and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

Parameters:

s - The encoded string.

Returns:

The decoded version.

6.717.2.2 static std::string decaf::internal::net::URIEncoderDecoder::encodeOthers (const std::string & *s*) [static]

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved. They are converted into their hexadecimal value prepended by ”.

For example: Euro currency symbol -> "%E2%82%AC".

Parameters:

s - the string to be converted

Returns:

the converted string

6.717.2.3 static std::string decaf::internal::net::URIEncoderDecoder::quoteIllegal (const std::string & *s*, const std::string & *legal*) [static]

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ”.

For example: '#' -> 23

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

Parameters:

s - the string to be converted

legal - the characters allowed to be preserved in the string *s*

Returns:

converted string

6.717.2.4 static void decaf::internal::net::URIEncoderDecoder::validate (const std::string & *s*, const std::string & *legal*) throw (decaf::net::URISyntaxException) [static]

Validate a string by checking if it contains any characters other than: 1. letters ('a'..'z', 'A'..'Z') 2. numbers ('0'..'9') 3. characters in the legalset parameter 4. characters that are not ISO Control or are not ISO Space characters)

Parameters:

s - the string to be validated

legal - the characters allowed in the string *s*

6.717.2.5 `static void decaf::internal::net::URIEncoderDecoder::validateSimple
(const std::string & s, const std::string & legal) throw (
decaf::net::URISyntaxException) [static]`

Validate a string by checking if it contains any characters other than: 1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9') 3. characters in the *legalset* parameter

Parameters:

- s* - the string to be validated
- legal* - the characters allowed in the string *s*

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIEncoderDecoder.h`

6.718 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)

*Setup the **URIHelper** (p. 3322) with values assigned to the various fields that are used in the validation process.*

- **URIHelper** ()

Sets up the filter strings with sane defaults.

- virtual ~**URIHelper** ()

- **URIType parseURI** (const std::string &uri, bool forceServer) throw (decaf::net::URISyntaxException)

Parse the passed in URI.

- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw (decaf::net::URISyntaxException)

Validate the schema portin of the URI.

- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Ssp Segment contains no invalid encodings.

- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Authority Segment contains no invalid encodings.

- void **validatePath** (const std::string &uri, const std::string &path, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Path Segment contains no invalid encodings.

- void **validateQuery** (const std::string &uri, const std::string &query, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Query Segment contains no invalid encodings.

- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI fragment contains no invalid encodings.

- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw (decaf::net::URISyntaxException)

determine the host, port and user-info if the authority parses successfully to a server based authority

- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size_t index) throw (decaf::net::URISyntaxException)
Check the supplied user info for validity.
- bool **isValidHost** (bool forceServer, const std::string &host) throw (decaf::net::URISyntaxException)
distinguish between IPv4, IPv6, domain name and validate it based on its type
- bool **isValidDomainName** (const std::string &host)
Validates the string past to determine if it is a well formed domain name.
- bool **isValidIPv4Address** (const std::string &host)
Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.
- bool **isValidIPv6Address** (const std::string &ipAddress)
Determines if the given address is valid according to the IPv6 spec.
- bool **isValidIPv4Word** (const std::string &word)
Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.
- bool **isValidHexChar** (char c)
Determines if the given char is a valid Hex char.

6.718.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

6.718.2 Constructor & Destructor Documentation

- #### 6.718.2.1 decaf::internal::net::URIHelper::URIHelper (const std::string & *unreserved*, const std::string & *punct*, const std::string & *reserved*, const std::string & *someLegal*, const std::string & *allLegal*)

Setup the **URIHelper** (p. 3322) with values assigned to the various fields that are used in the validation process. The defaults are overridden by these values.

Parameters:

- unreserved* - characters not reserved for use.
- punct* - allowable punctuation symbols.
- reserved* - characters not allowed for general use in the URI.
- someLegal* - characters that are legal in certain cases.
- allLegal* - characters that are always legal.

- #### 6.718.2.2 decaf::internal::net::URIHelper::URIHelper ()

Sets up the filter strings with sane defaults.

6.718.2.3 `virtual decaf::internal::net::URIHelper::~~URIHelper () [inline, virtual]`

6.718.3 Member Function Documentation

6.718.3.1 `bool decaf::internal::net::URIHelper::isValidDomainName (const std::string & host)`

Validates the string past to determine if it is a well formed domain name.

Parameters:

host - domain name to validate.

Returns:

true if host is well formed.

6.718.3.2 `bool decaf::internal::net::URIHelper::isValidHexChar (char c)`

Determines if the given char is a valid Hex char. Valid chars are A-F (upper or lower case) and 0-9.

Parameters:

c - char to inspect

Returns:

true if *c* is a valid hex char.

6.718.3.3 `bool decaf::internal::net::URIHelper::isValidHost (bool forceServer, const std::string & host) throw (decaf::net::URISyntaxException)`

distinguish between IPv4, IPv6, domain name and validate it based on its type

Parameters:

forceServer - true if the forceServer mode should be active.

host - Host string to validate.

Returns:

true if the host value if a valid domain name.

Exceptions:

URISyntaxException if the host is invalid and forceServer is true.

6.718.3.4 `bool decaf::internal::net::URIHelper::isValidIP4Word (const std::string & word)`

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

Parameters:

word - string value to check.

Returns:

true if the word is a valid IPv4 word.

6.718.3.5 `bool decaf::internal::net::URIHelper::isValidIPv6Address (const std::string & ipAddress)`

Determines if the given address is valid according to the IPv6 spec.

Parameters:

ipAddress - string ip address value to validate.

Returns:

true if the address string is valid.

6.718.3.6 `bool decaf::internal::net::URIHelper::isValidIPv4Address (const std::string & host)`

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9. and XXX is not greater than 255.

Parameters:

host - IPv4 address string to parse.

Returns:

true if host is a well formed IPv4 address.

6.718.3.7 `URIType decaf::internal::net::URIHelper::parseAuthority (bool forceServer, const std::string & authority) throw (decaf::net::URISyntaxException)`

determine the host, port and user-info if the authority parses successfully to a server based authority behavior in error cases: if forceServer is true, throw URISyntaxException with the proper diagnostic messages. if forceServer is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the forceServer parameter e.g. mal-formed ipv6 address

Parameters:

forceServer

authority

Returns:

a **URIType** (p. 3339) instance containing the parsed data.

Exceptions:

URISyntaxException

6.718.3.8 `URIType decaf::internal::net::URIHelper::parseURI (const std::string & uri, bool forceServer) throw (decaf::net::URISyntaxException)`

Parse the passed in URI.

Parameters:

uri - the URI to Parse

forceServer - if true invalid URI data throws an Exception

Returns:

a `URIType` (p. 3339) instance containing the parsed data.

Exceptions:

URISyntaxException if *forceServer* is true and the URI is invalid.

6.718.3.9 `void decaf::internal::net::URIHelper::validateAuthority (const std::string & uri, const std::string & authority, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Authority Segment contains no invalid encodings.

Parameters:

uri - the full uri.

authority - the Authority to check.

index - position in the uri where Authority starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.718.3.10 `void decaf::internal::net::URIHelper::validateFragment (const std::string & uri, const std::string & fragment, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI fragment contains no invalid encodings.

Parameters:

uri - the full uri.

fragment - the fragment to check.

index - position in the uri where fragment starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.718.3.11 `void decaf::internal::net::URIHelper::validatePath (const std::string & uri, const std::string & path, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Path Segment contains no invalid encodings.

Parameters:

uri - the full uri.

path - the path to check.

index - position in the uri where path starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.718.3.12 `void decaf::internal::net::URIHelper::validateQuery (const std::string & uri, const std::string & query, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Query Segment contains no invalid encodings.

Parameters:

uri - the full uri.

query - the query to check.

index - position in the uri where fragment starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.718.3.13 `void decaf::internal::net::URIHelper::validateScheme (const std::string & uri, const std::string & scheme, int index) throw (decaf::net::URISyntaxException)`

Validate the schema portin of the URI.

Parameters:

uri - the URI to check.

scheme - the schema section of the URI.

index - index in uri where schema starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.718.3.14 void decaf::internal::net::URIHelper::validateSsp (const std::string & *uri*, const std::string & *ssp*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Ssp Segment contains no invalid encodings.

Parameters:

uri - the full uri.

ssp - the SSP to check.

index - position in the uri where Ssp starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.718.3.15 void decaf::internal::net::URIHelper::validateUserinfo (const std::string & *uri*, const std::string & *userinfo*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Check the supplied user info for validity.

Parameters:

uri - the uri to parse.

userinfo - supplied user info

index - index into the URI string where the data is located.

Returns:

true if valid

Exceptions:

URISyntaxException if an error occurs

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIHelper.h**

6.719 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

Public Member Functions

- **URIPool** ()
Create an Empty URI Pool.
- **URIPool** (const **decaf::util::List**< **URI** > &uris)
Creates a new URI Pool using the given list as the initial Free List.
- virtual ~**URIPool** ()
- **URI** **getURI** () throw (**decaf::lang::exceptions::NoSuchElementException**)
*Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a **NoSuchElementException**.*
- void **addURI** (const **URI** &uri)
*Adds a URI to the free list, callers that have previously taken one using the **getURI** method should always return the URI when they close the resource that was connected to that URI.*
- void **addURIs** (const **StlList**< **URI** > &uris)
Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.
- void **removeURI** (const **URI** &uri)
Remove a given URI from the Free List.
- bool **isRandomize** () const
Is the URI that is given randomly picked from the pool or is each one taken in sequence.
- void **setRandomize** (bool value)
Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

6.719.1 Constructor & Destructor Documentation

6.719.1.1 activemq::transport::failover::URIPool::URIPool ()

Create an Empty URI Pool.

6.719.1.2 activemq::transport::failover::URIPool::URIPool (const **decaf::util::List**< **URI** > & *uris*)

Creates a new URI Pool using the given list as the initial Free List.

Parameters:

uris - List of URI to place in the Pool.

6.719.1.3 `virtual activemq::transport::failover::URIPool::~~URIPool ()` [virtual]

6.719.2 Member Function Documentation

6.719.2.1 `void activemq::transport::failover::URIPool::addURI (const URI & uri)`

Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

Parameters:

uri - a URI previously taken from the pool.

6.719.2.2 `void activemq::transport::failover::URIPool::addURIs (const StlList< URI > & uris)`

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

Parameters:

uris - List of URIs to add into the Pool.

6.719.2.3 `URI activemq::transport::failover::URIPool::getURI () throw (decaf::lang::exceptions::NoSuchElementException)`

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a `NoSuchElementException`. Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

Returns:

the next free URI in the Pool.

Exceptions:

NoSuchElementException if there are none free currently.

6.719.2.4 `bool activemq::transport::failover::URIPool::isRandomize () const [inline]`

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

Returns:

true if URI gets are random.

6.719.2.5 `void activemq::transport::failover::URIPool::removeURI (const URI & uri)`

Remove a given URI from the Free List.

Parameters:

uri - the URI to find and remove from the free list

6.719.2.6 void activemq::transport::failover::URIPool::setRandomize (bool *value*)
[inline]

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

Parameters:

value - true indicates URI gets are random.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/URIPool.h`

6.720 activemq::util::URISupport Class Reference

```
#include <src/main/activemq/util/URISupport.h>
```

Static Public Member Functions

- static void **parseURL** (const std::string &URI, decaf::util::Properties &properties) throw (decaf::lang::exceptions::IllegalArgumentException)
Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.
- static **CompositeData** **parseComposite** (const URI &uri) throw (decaf::net::URISyntaxException)
Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.
- static decaf::util::Properties **parseQuery** (std::string query) throw (decaf::lang::exceptions::IllegalArgumentException)
Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.
- static void **parseQuery** (std::string query, decaf::util::Properties *properties) throw (decaf::lang::exceptions::IllegalArgumentException)
Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.
- static std::string **createQueryString** (const Properties &options) throw (decaf::net::URISyntaxException)
Given a properties object create a string that can be appended to a URI as a valid Query string.

6.720.1 Member Function Documentation

6.720.1.1 static std::string activemq::util::URISupport::createQueryString (const Properties & *options*) throw (decaf::net::URISyntaxException)
 [static]

Given a properties object create a string that can be appended to a URI as a valid Query string.

Parameters:

options Properties object containing key / value query values.

Returns:

a valid URI query string.

Exceptions:

URISyntaxException if the string in the Properties object can't be encoded into a valid URI Query string.

6.720.1.2 static `CompositeData` `activemq::util::URISupport::parseComposite`
(const URI & *uri*) throw (`decaf::net::URISyntaxException`) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

Parameters:

uri - The Composite URI to parse.

Returns:

a new `CompositeData` (p.1088) object with the parsed data

Exceptions:

URISyntaxException if the URI is not well formed.

6.720.1.3 static void `activemq::util::URISupport::parseQuery` (std::string
query, `decaf::util::Properties` * *properties*) throw (`decaf::lang::exceptions::IllegalArgumentException`) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters:

query - the query string to parse.

properties - object pointer to get the parsed output.

Exceptions:

IllegalArgumentException if the Query string is not well formed.

6.720.1.4 static `decaf::util::Properties` `activemq::util::URISupport::parseQuery` (std::string *query*)
throw (`decaf::lang::exceptions::IllegalArgumentException`) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters:

query The query string to parse and extract the encoded properties.

Returns:

Properties object with the parsed output.

Exceptions:

IllegalArgumentException if the Query string is not well formed.

6.720.1.5 static void activemq::util::URISupport::parseURL (const std::string & *URI*, decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

Parameters:

URI a Broker URI to parse

properties a Properties object to set the parsed values in

Exceptions:

IllegalArgumentException if the passed URI is invalid

The documentation for this class was generated from the following file:

- src/main/activemq/util/**URISupport.h**

6.721 decaf::net::URISyntaxException Class Reference

#include <src/main/decaf/net/URISyntaxException.h> Inheritance diagram for decaf::net::URISyntaxException:

Public Member Functions

- **URISyntaxException** () throw ()
Default Constructor.
- **URISyntaxException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()
Copy Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const std::exception *cause) throw ()
Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const char *msg DECAF_ - UNUSED) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason, std::size_t index) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **URISyntaxException** * **clone** () const
Clones this exception.
- virtual ~**URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- std::size_t **getIndex** () const

6.721.1 Constructor & Destructor Documentation

6.721.1.1 decaf::net::URISyntaxException::URISyntaxException () throw () [inline]

Default Constructor.

6.721.1.2 decaf::net::URISyntaxException::URISyntaxException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.721.1.3 decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.721.1.4 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.721.1.5 decaf::net::URISyntaxException::URISyntaxException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.721.1.6 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const char **msg* *DECAF_UNUSED*) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.721.1.7 `decaf::net::URISyntaxException::URISyntaxException (const char * file,
const int lineNumber, const std::string & input, const std::string &
reason) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the input string that caused the error and the reason for the error.

Parameters:

file The file name where exception occurs.
lineNumber The line number where the exception occurred.
input The **URL** (p.3347) that caused the exception.
reason The reason for the failure.

6.721.1.8 `decaf::net::URISyntaxException::URISyntaxException (const char * file,
const int lineNumber, const std::string & input, const std::string &
reason, std::size_t index) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the input string that caused the error and the reason for the error.

Parameters:

file The file name where exception occurs
lineNumber The line number where the exception occurred.
input The input **URI** (p.3308) that caused the exception
reason The reason for the failure.
index The index in the **URI** (p.3308) string where the error occurred.

6.721.1.9 `virtual decaf::net::URISyntaxException::~~URISyntaxException () throw
()` [inline, virtual]

6.721.2 Member Function Documentation

6.721.2.1 `virtual URISyntaxException* decaf::net::URISyntaxException::clone ()
const` [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p.1577).

6.721.2.2 `std::size_t decaf::net::URISyntaxException::getIndex () const` [inline]

Returns:

the index in the input string where the error occurred or -1

6.721.2.3 `std::string decaf::net::URISyntaxException::getInput () const` [inline]

Returns:

the Input string that cause this exception or ""

6.721.2.4 `std::string decaf::net::URISyntaxException::getReason () const` [inline]

Returns:

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`

6.722 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const
*Gets the source URI string that was parsed to obtain this **URIType** (p. 3339) instance and the resulting data,.*
- void **setSource** (const std::string &source)
*Sets the source URI string that was parsed to obtain this **URIType** (p. 3339) instance and the resulting data,.*
- std::string **getScheme** () const
Gets the Scheme of the URI, e.g.
- void **setScheme** (const std::string &scheme)
Sets the Scheme of the URI, e.g.
- std::string **getSchemeSpecificPart** () const
Gets the Scheme Specific Part of the URI.
- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)
Sets the Scheme Specific Part of the URI.
- std::string **getAuthority** () const
Gets the Authority of the URI.
- void **setAuthority** (const std::string &authority)
Sets the Authority of the URI.
- std::string **getUserInfo** () const
Gets the user info part of the URI, e.g.
- void **setUserInfo** (const std::string &userinfo)
Sets the user info part of the URI, e.g.
- std::string **getHost** () const
Gets the Host name part of the URI.
- void **setHost** (const std::string &host)
Sets the Host name part of the URI.
- int **getPort** () const

Gets the port part of the URI.

- void **setPort** (int port)
Sets the port part of the URI.
- std::string **getPath** () const
Gets the Path part of the URI.
- void **setPath** (const std::string &path)
Sets the Path part of the URI.
- std::string **getQuery** () const
Gets the Query part of the URI.
- void **setQuery** (const std::string &query)
Sets the Query part of the URI.
- std::string **getFragment** () const
Gets the Fragment part of the URI.
- void **setFragment** (const std::string &fragment)
Sets the Fragment part of the URI.
- bool **isOpaque** () const
Gets if the URI is Opaque.
- void **setOpaque** (bool opaque)
Sets if the URI is Opaque.
- bool **isAbsolute** () const
Gets if the URI is Absolute.
- void **setAbsolute** (bool absolute)
Sets if the URI is Absolute.
- bool **isServerAuthority** () const
Gets if the URI is a Server Authority.
- void **setServerAuthority** (bool serverAuthority)
Sets if the URI is a Server Authority.
- bool **isValid** () const
Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.
- void **setValid** (bool valid)
Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

6.722.1 Detailed Description

Basic type object that holds data that composes a given URI.

6.722.2 Constructor & Destructor Documentation

6.722.2.1 `decaf::internal::net::URIType::URIType (const std::string & source)` `[inline]`

6.722.2.2 `decaf::internal::net::URIType::URIType ()` `[inline]`

6.722.2.3 `virtual decaf::internal::net::URIType::~~URIType ()` `[inline, virtual]`

6.722.3 Member Function Documentation

6.722.3.1 `std::string decaf::internal::net::URIType::getAuthority () const` `[inline]`

Gets the Authority of the URI.

Returns:

Authority part string.

6.722.3.2 `std::string decaf::internal::net::URIType::getFragment () const` `[inline]`

Gets the Fragment part of the URI.

Returns:

Fragment part string.

6.722.3.3 `std::string decaf::internal::net::URIType::getHost () const` `[inline]`

Gets the Host name part of the URI.

Returns:

Host name part string.

6.722.3.4 `std::string decaf::internal::net::URIType::getPath () const` `[inline]`

Gets the Path part of the URI.

Returns:

Path part string.

6.722.3.5 `int decaf::internal::net::URIType::getPort () const [inline]`

Gets the port part of the URL.

Returns:

port part string, -1 if not set.

6.722.3.6 `std::string decaf::internal::net::URIType::getQuery () const [inline]`

Gets the Query part of the URL.

Returns:

Query part string.

6.722.3.7 `std::string decaf::internal::net::URIType::getScheme () const [inline]`

Gets the Scheme of the URL, e.g. scheme ("http"/"ftp"/...).

Returns:

scheme part string.

6.722.3.8 `std::string decaf::internal::net::URIType::getSchemeSpecificPart () const [inline]`

Gets the Scheme Specific Part of the URL.

Returns:

scheme specific part string.

6.722.3.9 `std::string decaf::internal::net::URIType::getSource () const [inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p.3339) instance and the resulting data,.

Returns:

the source URI string

6.722.3.10 `std::string decaf::internal::net::URIType::getUserInfo () const [inline]`

Gets the user info part of the URL, e.g. user name, as in http://user:passwd@host:port/

Returns:

user info part string.

6.722.3.11 `bool decaf::internal::net::URIType::isAbsolute () const [inline]`

Gets if the URI is Absolute.

Returns:

true if Absolute.

6.722.3.12 `bool decaf::internal::net::URIType::isOpaque () const [inline]`

Gets if the URI is Opaque.

Returns:

true if opaque.

6.722.3.13 `bool decaf::internal::net::URIType::isServerAuthority () const [inline]`

Gets if the URI is a Server Authority.

Returns:

true if Server Authority.

6.722.3.14 `bool decaf::internal::net::URIType::isValid () const [inline]`

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Returns:

true if the **URIType** (p. 3339) contains valid data.

6.722.3.15 `void decaf::internal::net::URIType::setAbsolute (bool absolute) [inline]`

Sets if the URI is Absolute.

Parameters:

absolute - true if Absolute.

6.722.3.16 `void decaf::internal::net::URIType::setAuthority (const std::string &authority) [inline]`

Sets the Authority of the URI.

Parameters:

authority Authority part string.

6.722.3.17 void decaf::internal::net::URIType::setFragment (const std::string & *fragment*) [inline]

Sets the Fragment part of the URI.

Parameters:

fragment - Fragment part string.

6.722.3.18 void decaf::internal::net::URIType::setHost (const std::string & *host*) [inline]

Sets the Host name part of the URI.

Parameters:

host - Host name part string.

6.722.3.19 void decaf::internal::net::URIType::setOpaque (bool *opaque*) [inline]

Sets if the URI is Opaque.

Parameters:

opaque true if opaque.

6.722.3.20 void decaf::internal::net::URIType::setPath (const std::string & *path*) [inline]

Sets the Path part of the URI.

Parameters:

path - Path part string.

6.722.3.21 void decaf::internal::net::URIType::setPort (int *port*) [inline]

Sets the port part of the URI.

Parameters:

port - port part string, -1 if not set.

6.722.3.22 void decaf::internal::net::URIType::setQuery (const std::string & *query*) [inline]

Sets the Query part of the URI.

Parameters:

query - Query part string.

6.722.3.23 `void decaf::internal::net::URIType::setScheme (const std::string & scheme)` [inline]

Sets the Scheme of the URI, e.g. `scheme ("http"/"ftp"/...)`.

Parameters:

scheme - scheme part string.

6.722.3.24 `void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & schemeSpecificPart)` [inline]

Sets the Scheme Specific Part of the URI.

Parameters:

schemeSpecificPart - scheme specific part string.

6.722.3.25 `void decaf::internal::net::URIType::setServerAuthority (bool serverAuthority)` [inline]

Sets if the URI is a Server Authority.

Parameters:

serverAuthority - true if Server Authority.

6.722.3.26 `void decaf::internal::net::URIType::setSource (const std::string & source)` [inline]

Sets the source URI string that was parsed to obtain this **URIType** (p. 3339) instance and the resulting data,.

Parameters:

source - the source URI string

6.722.3.27 `void decaf::internal::net::URIType::setUserInfo (const std::string & userinfo)` [inline]

Sets the user info part of the URI, e.g. user name, as in `http://user:passwd@host:port/`

Parameters:

userinfo - user info part string.

6.722.3.28 `void decaf::internal::net::URIType::setValid (bool valid)` [inline]

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Parameters:

valid - true if the **URIType** (p. 3339) contains valid data.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIType.h**

6.723 decaf::net::URL Class Reference

Class **URL** (p. 3347) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

```
#include <src/main/decaf/net/URL.h>
```

Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

6.723.1 Detailed Description

Class **URL** (p. 3347) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

```
http://www.ksc.nasa.gov/facts/internet/url-primer.html
```

In general, a **URL** (p. 3347) can be broken into several parts. The previous example of a **URL** (p. 3347) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named www.ksc.nasa.gov. The information on that host machine is named /facts/internet/url-primer.html. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 3347) is called the path component.

A **URL** (p. 3347) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

```
http://www.ksc.nasa.gov:80/facts/internet/url-primer.html
```

The syntax of **URL** (p. 3347) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 3308)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports scope_ids. The syntax and usage of scope_ids is described here.

A **URL** (p. 3347) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

```
http://www.apache.org/cms/index.html#chapter1
```

This fragment is not technically part of the **URL** (p. 3347). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag chapter1 attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 3347). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 3347):

```
http://www.apache.org/cms/index.html
```


contained within it the relative **URL** (p. 3347):

FAQ.html

it would be a shorthand for:

`http://www.apache.org/cms/FAQ.html`

The relative **URL** (p. 3347) need not specify all the components of a **URL** (p. 3347). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 3347). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 3347) class does not itself encode or decode any **URL** (p. 3347) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 3347), and also to decode any escaped fields, that are returned from **URL** (p. 3347). Furthermore, because **URL** (p. 3347) has no knowledge of **URL** (p. 3347) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 3347). For example, the two URLs:

`http://foo.com/hello world/` and `http://foo.com/hello%20world`

would be considered not equal to each other.

Note, the **URI** (p. 3308) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 3308), and to convert between these two classes using `toURI()` and `URI.toURL()` (p. 3318).

The **URLEncoder** (p. 3350) and **URLDecoder** (p. 3349) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

6.723.2 Constructor & Destructor Documentation

6.723.2.1 decaf::net::URL::URL ()

6.723.2.2 decaf::net::URL::URL (const std::string & url)

6.723.2.3 virtual decaf::net::URL::~~URL () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URL.h`

6.724 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

Public Member Functions

- virtual `~URLDecoder ()`

Static Public Member Functions

- static `std::string decode (const std::string &value)`
Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type.

6.724.1 Constructor & Destructor Documentation

6.724.1.1 `virtual decaf::net::URLDecoder::~~URLDecoder ()` [inline, virtual]

6.724.2 Member Function Documentation

6.724.2.1 `static std::string decaf::net::URLDecoder::decode (const std::string &value)` [static]

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type. '+' will be converted to space, '%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

Parameters:

value - string The encoded string.

Returns:

The decoded version as a string.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLDecoder.h`

6.725 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

Public Member Functions

- virtual `~URLEncoder()`

Static Public Member Functions

- static `std::string encode(const std::string &value)`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

6.725.1 Constructor & Destructor Documentation

6.725.1.1 virtual `decaf::net::URLEncoder::~~URLEncoder()` [inline, virtual]

6.725.2 Member Function Documentation

6.725.2.1 static `std::string decaf::net::URLEncoder::encode(const std::string &value)` [static]

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type. All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '*', '_' are converted into their hexadecimal value prepended by ".

For example: '#' -> 23

In addition, spaces are substituted by '+'

Parameters:

value - the string to be converted

Returns:

the converted string

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLEncoder.h`

6.726 activemq::util::Usage Class Reference

#include <src/main/activemq/util/Usage.h> Inheritance diagram for activemq::util::Usage:

Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`
*Waits forever for more space to be returned to this **Usage** (p. 3351) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`
*Waits for more space to be returned to this **Usage** (p. 3351) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void `increaseUsage (unsigned long long value)=0`
Increases the usage by the value amount.
- virtual void `decreaseUsage (unsigned long long value)=0`
Decreases the usage by the value amount.
- virtual bool `isFull () const =0`
*Returns true if this **Usage** (p. 3351) instance is full, i.e.*

6.726.1 Constructor & Destructor Documentation

6.726.1.1 virtual `activemq::util::Usage::~~Usage ()` [inline, virtual]

6.726.2 Member Function Documentation

6.726.2.1 virtual void `activemq::util::Usage::decreaseUsage (unsigned long long value)` [pure virtual]

Decreases the usage by the value amount.

Parameters:

value Amount of space to return to the pool

Implemented in `activemq::util::MemoryUsage` (p. 2142).

6.726.2.2 virtual void `activemq::util::Usage::enqueueUsage (unsigned long long value)` [pure virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters:

value Amount of usage in bytes to add.

Implemented in **activemq::util::MemoryUsage** (p. 2142).

6.726.2.3 virtual void activemq::util::Usage::increaseUsage (unsigned long long *value*) [pure virtual]

Increases the usage by the value amount.

Parameters:

value Amount of usage to add.

Implemented in **activemq::util::MemoryUsage** (p. 2143).

6.726.2.4 virtual bool activemq::util::Usage::isFull () const [pure virtual]

Returns true if this **Usage** (p. 3351) instance is full, i.e. **Usage** (p. 3351) $\geq 100\%$

Returns:

true if **Usage** (p. 3351) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 2143).

6.726.2.5 virtual void activemq::util::Usage::waitForSpace (unsigned int *timeout*) [pure virtual]

Waits for more space to be returned to this **Usage** (p. 3351) Manager, times out when the given time span in milliseconds elapses.

Parameters:

timeout The time to wait for more space.

Implemented in **activemq::util::MemoryUsage** (p. 2143).

6.726.2.6 virtual void activemq::util::Usage::waitForSpace () [pure virtual]

Waits forever for more space to be returned to this **Usage** (p. 3351) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 2144).

The documentation for this class was generated from the following file:

- src/main/activemq/util/Usage.h

6.727 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

#include <src/main/decaf/io/UTFDataFormatException.h>Inheritance diagram for decaf::io::UTFDataFormatException:

Public Member Functions

- **UTFDataFormatException** () throw ()
Default Constructor.
- **UTFDataFormatException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **UTFDataFormatException** (const UTFDataFormatException &ex) throw ()
Copy Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UTFDataFormatException** (const std::exception *cause) throw ()
Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UTFDataFormatException * clone** () const
Clones this exception.
- virtual **~UTFDataFormatException** () throw ()

6.727.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since:

1.0

6.727.2 Constructor & Destructor Documentation

6.727.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException () throw () [inline]

Default Constructor.

6.727.2.2 decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.727.2.3 decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.727.2.4 decaf::io::UTFDataFormatException::UTFDataFormatException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.727.2.5 decaf::io::UTFDataFormatException::UTFDataFormatException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.727.2.6 decaf::io::UTFDataFormatException::UTFDataFormatException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.727.2.7 `virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException
() throw () [inline, virtual]`

6.727.3 Member Function Documentation

6.727.3.1 `virtual UTFDataFormatException* de-
caf::io::UTFDataFormatException::clone () const
[inline, virtual]`

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 1822).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UTFDataFormatException.h`

6.728 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 3356)).

#include <src/main/decaf/util/UUID.h> Inheritance diagram for decaf::util::UUID:

Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)
*Constructs a new **UUID** (p. 3356) using the specified data.*
- virtual **~UUID** ()
- virtual int **compareTo** (const **UUID** &value) const
*Compare the given **UUID** (p. 3356) to this one.*
- virtual bool **equals** (const **UUID** &value) const
*Compares this **UUID** (p. 3356) to the one given, returns true if they are equal.*
- virtual bool **operator==** (const **UUID** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **UUID** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual std::string **toString** () const
*Returns a String object representing this **UUID** (p. 3356).*
- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw (lang::exceptions::UnsupportedOperationException)
*The node value associated with this **UUID** (p. 3356).*
- virtual long long **timestamp** () throw (lang::exceptions::UnsupportedOperationException)
*The timestamp value associated with this **UUID** (p. 3356).*
- virtual int **clockSequence** () throw (lang::exceptions::UnsupportedOperationException)
*The clock sequence value associated with this **UUID** (p. 3356).*
- virtual int **variant** () throw (lang::exceptions::UnsupportedOperationException)
*The variant number associated with this **UUID** (p. 3356).*
- virtual int **version** () throw (lang::exceptions::UnsupportedOperationException)
*The version number associated with this **UUID** (p. 3356).*

Static Public Member Functions

- static **UUID** **randomUUID** ()
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3356).*
- static **UUID** **nameUUIDFromBytes** (const std::vector< char > &name)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3356) based on the specified byte array.*
- static **UUID** **nameUUIDFromBytes** (const char *name, std::size_t size)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3356) based on the specified byte array.*
- static **UUID** **fromString** (const std::string &name) throw (lang::exceptions::IllegalArgumentException)
*Creates a **UUID** (p. 3356) from the string standard representation as described in the **toString()** (p. 3361) method.*

6.728.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 3356)). A **UUID** (p. 3356) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 3356) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 3356) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFFFF00 time_mid 0x000000000000F000 version
0x00000000000000FF time_hi
```

The least significant long consists of the following unsigned fields:

```
0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFF node
```

The variant field contains a value which identifies the layout of the **UUID** (p. 3356). The bit layout described above is valid only for a **UUID** (p. 3356) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 3356). There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

6.728.2 Constructor & Destructor Documentation

6.728.2.1 decaf::util::UUID::UUID (long long *mostSigBits*, long long *leastSigBits*)

Constructs a new **UUID** (p. 3356) using the specified data. *mostSigBits* is used for the most significant 64 bits of the **UUID** (p. 3356) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 3356).

Parameters:*mostSigBits**leastSigBits***6.728.2.2** virtual decaf::util::UUID::~~UUID () [virtual]**6.728.3 Member Function Documentation****6.728.3.1** virtual int decaf::util::UUID::clockSequence () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The clock sequence value associated with this **UUID** (p. 3356). The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 3356). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 3356).

The clockSequence value is only meaningful in a time-based **UUID** (p. 3356), which has version type 1. If this **UUID** (p. 3356) is not a time-based **UUID** (p. 3356) then this method throws UnsupportedOperationException.

Returns:

the clockSequeunce associated with a V1 **UUID** (p. 3356)

Exceptions:*UnsupportedOperationException***6.728.3.2** virtual int decaf::util::UUID::compareTo (const UUID & *value*) const [virtual]

Compare the given **UUID** (p. 3356) to this one.

Parameters:

value - the **UUID** (p. 3356) to compare to

6.728.3.3 virtual bool decaf::util::UUID::equals (const UUID & *value*) const [virtual]

Compares this **UUID** (p. 3356) to the one given, returns true if they are equal.

Parameters:

value - the **UUID** (p. 3356) to compare to.

Returns:

true if UUIDs are the same.

6.728.3.4 `static UUID decaf::util::UUID::fromString (const std::string & name)
throw (lang::exceptions::IllegalArgumentException) [static]`

Creates a **UUID** (p. 3356) from the string standard representation as described in the `toString()` (p. 3361) method.

Parameters:

name - a string to be used to construct a **UUID** (p. 3356).

Returns:

type 3 **UUID** (p. 3356)

6.728.3.5 `virtual long long decaf::util::UUID::getLeastSignificantBits () const
[virtual]`

Returns:

the most significant 64 bits of this **UUID**'s 128 bit value.

6.728.3.6 `virtual long long decaf::util::UUID::getMostSignificantBits () const
[virtual]`

Returns:

the most significant 64 bits of this **UUID**'s 128 bit value.

6.728.3.7 `static UUID decaf::util::UUID::nameUUIDFromBytes (const char *
name, std::size_t size) [static]`

Static factory to retrieve a type 3 (name based) **UUID** (p. 3356) based on the specified byte array.

Parameters:

name - a byte array to be used to construct a **UUID** (p. 3356).

size - the size of the byte array, or number of bytes to use.

Returns:

type 3 **UUID** (p. 3356)

6.728.3.8 `static UUID decaf::util::UUID::nameUUIDFromBytes (const std::vector<
char > & name) [static]`

Static factory to retrieve a type 3 (name based) **UUID** (p. 3356) based on the specified byte array.

Parameters:

name - a byte array to be used to construct a **UUID** (p. 3356).

Returns:

type 3 **UUID** (p. 3356)

6.728.3.9 `virtual long long decaf::util::UUID::node () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The node value associated with this **UUID** (p. 3356). The 48 bit node value is constructed from the node field of this **UUID** (p. 3356). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 3356) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 3356), which has version type 1. If this **UUID** (p. 3356) is not a time-based **UUID** (p. 3356) then this method throws `UnsupportedOperationException`.

Returns:

the node value of this **UUID** (p. 3356)

Exceptions:

UnsupportedOperationException

6.728.3.10 `virtual bool decaf::util::UUID::operator< (const UUID & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.728.3.11 `virtual bool decaf::util::UUID::operator== (const UUID & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.728.3.12 `static UUID decaf::util::UUID::randomUUID () [static]`

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3356). The **UUID** (p. 3356) is generated using a cryptographically strong pseudo random number generator.

Returns:

type 4 **UUID** (p. 3356)

6.728.3.13 `virtual long long decaf::util::UUID::timestamp () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The timestamp value associated with this **UUID** (p. 3356). The 60 bit timestamp value is constructed from the `time_low`, `time_mid`, and `time_hi` fields of this **UUID** (p. 3356). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 3356), which has version type 1. If this **UUID** (p. 3356) is not a time-based **UUID** (p. 3356) then this method throws `UnsupportedOperationException`.

Returns:

the timestamp associated with a V1 **UUID** (p. 3356)

Exceptions:

UnsupportedOperationException

6.728.3.14 `virtual std::string decaf::util::UUID::toString () const [virtual]`

Returns a `String` object representing this **UUID** (p. 3356). **UUID**'s are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

Returns:

formatted string for this **UUID** (p. 3356)

6.728.3.15 `virtual int decaf::util::UUID::variant () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The variant number associated with this **UUID** (p. 3356). The variant number describes the layout of the **UUID** (p. 3356). The variant number has the following meaning:

* 0 Reserved for NCS backward compatibility * 2 The Leach-Salz variant (used by this class) * 6 Reserved, Microsoft Corporation backward compatibility * 7 Reserved for future definition

Returns:

the variant associated with a V1 **UUID** (p. 3356)

Exceptions:

UnsupportedOperationException

6.728.3.16 `virtual int decaf::util::UUID::version () throw (lang::exceptions::UnsupportedOperationException) [virtual]`

The version number associated with this **UUID** (p. 3356). The version number describes how this **UUID** (p. 3356) was generated. The version number has the following meaning:

* 1 Time-based **UUID** (p. 3356) * 2 DCE security **UUID** (p. 3356) * 3 Name-based **UUID** (p. 3356) * 4 Randomly generated **UUID** (p. 3356)

Returns:

the version associated with a V1 **UUID** (p. 3356)

Exceptions:

UnsupportedOperationException

The documentation for this class was generated from the following file:

- src/main/decaf/util/**UUID.h**

6.729 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

#include <src/main/activemq/wireformat/WireFormat.h> Inheritance diagram for activemq::wireformat::WireFormat:

Public Member Functions

- virtual `~WireFormat ()`
- virtual void `marshal (const Pointer< commands::Command > &command, const activemq::transport::Transport *transport, decaf::io::DataOutputStream *out)=0`
throw (decaf::io::IOException)
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- virtual `Pointer< commands::Command > unmarshal (const activemq::transport::Transport *transport, decaf::io::DataInputStream *in)=0`
throw (decaf::io::IOException)
Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.
- virtual void `setVersion (int version)=0`
Set the Version.
- virtual int `getVersion () const =0`
Get the Version.
- virtual bool `hasNegotiator () const =0`
Returns true if this WireFormat (p. 3363) has a Negotiator that needs to wrap the Transport that uses it.
- virtual bool `inReceive () const =0`
Indicates if the WireFormat object is in the process of receiving a message.
- virtual `Pointer< transport::Transport > createNegotiator (const Pointer< transport::Transport > &transport)=0` throw (decaf::lang::exceptions::UnsupportedOperationException)
If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

6.729.1 Detailed Description

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Version:

Revision

1.1

6.729.2 Constructor & Destructor Documentation

6.729.2.1 virtual `activemq::wireformat::WireFormat::~WireFormat ()` [inline, virtual]

6.729.3 Member Function Documentation

6.729.3.1 virtual `Pointer<transport::Transport> activemq::wireformat::WireFormat::createNegotiator (const Pointer< transport::Transport > & transport) throw (decaf::lang::exceptions::UnsupportedOperationException)` [pure virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters:

transport - the Transport to Wrap the Negotiator around.

Returns:

new instance of a **WireFormatNegotiator** (p. 3399) as a **Pointer<Transport>** (p. 2497).

Exceptions:

UnsupportedOperationException if the **WireFormat** (p. 3363) doesn't have a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2451), and **activemq::wireformat::stomp::StompWireFormat** (p. 3072).

6.729.3.2 virtual `int activemq::wireformat::WireFormat::getVersion ()` const [pure virtual]

Get the Version.

Returns:

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2453), and **activemq::wireformat::stomp::StompWireFormat** (p. 3072).

6.729.3.3 virtual `bool activemq::wireformat::WireFormat::hasNegotiator ()` const [pure virtual]

Returns true if this **WireFormat** (p. 3363) has a Negotiator that needs to wrap the Transport that uses it.

Returns:

true if the **WireFormat** (p. 3363) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2453), and **activemq::wireformat::stomp::StompWireFormat** (p. 3072).

6.729.3.4 virtual bool activemq::wireformat::WireFormat::inReceive () const [pure virtual]

Indicates if the WireFormat object is in the process of receiving a message. This is useful for monitoring inactivity and the **WireFormat** (p. 3363) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 3363) instance to determine if its busy or not and not mark the connection as inactive if so.

Returns:

true if the **WireFormat** (p. 3363) object is unmarshaling a message.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2453), and **activemq::wireformat::stomp::StompWireFormat** (p. 3073).

6.729.3.5 virtual void activemq::wireformat::WireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport * *transport*, decaf::io::DataOutputStream * *out*) throw (decaf::io::IOException) [pure virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters:

command The Command to Marshal
transport The Transport that called this method.
out The output stream to write the command to.

Exceptions:

IOException

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2455), and **activemq::wireformat::stomp::StompWireFormat** (p. 3073).

6.729.3.6 virtual void activemq::wireformat::WireFormat::setVersion (int *version*) [pure virtual]

Set the Version.

Parameters:

version the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2457), and **activemq::wireformat::stomp::StompWireFormat** (p. 3073).

6.729.3.7 virtual Pointer<commands::Command> activemq::wireformat::WireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form. Returns a Pointer to the newly unmarshaled Command.

Parameters:

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns:

the newly marshaled Command, caller owns the pointer

Exceptions:

IOException

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2459), and **activemq::wireformat::stomp::StompWireFormat** (p. 3073).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

6.730 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 3367) is the interface that all **WireFormatFactory** (p. 3367) classes must extend.

#include <src/main/activemq/wireformat/WireFormatFactory.h> Inheritance diagram for activemq::wireformat::WireFormatFactory:

Public Member Functions

- virtual **~WireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties)=0 throw (**decaf::lang::exceptions::IllegalStateException**)
*Creates a new **WireFormat** (p. 3363) Object passing it a set of properties from which it can obtain any optional settings.*

6.730.1 Detailed Description

The **WireFormatFactory** (p. 3367) is the interface that all **WireFormatFactory** (p. 3367) classes must extend. The Factory creates a **WireFormat** (p. 3363) Object based on the properties that are set in the passed **Properties** object.

6.730.2 Constructor & Destructor Documentation

6.730.2.1 virtual **activemq::wireformat::WireFormatFactory::~~WireFormatFactory** () [inline, virtual]

6.730.3 Member Function Documentation

6.730.3.1 virtual **Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) throw (**decaf::lang::exceptions::IllegalStateException**) [pure virtual]

Creates a new **WireFormat** (p. 3363) Object passing it a set of properties from which it can obtain any optional settings.

Parameters:

properties - the **Properties** for this **WireFormat** (p. 3363)

Returns:

Pointer to a new instance of a **WireFormat** (p. 3363) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 2460), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 3075).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatFactory.h`

6.731 activemq::commands::WireFormatInfo Class Reference

#include <src/main/activemq/commands/WireFormatInfo.h> Inheritance diagram for activemq::commands::WireFormatInfo:

Public Member Functions

- **WireFormatInfo** ()
- virtual **~WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataStructure** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1461) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- int **getVersion** () const
Get the current Wireformat Version.
- void **setVersion** (int version)
Set the current Wireformat Version.
- long long **getMaxInactivityDuration** () const
Returns the currently configured Max Inactivity duration.
- void **setMaxInactivityDuration** (long long maxInactivityDuration)
Sets the Max inactivity duration value.

- long long **getMaxInactivityDurationInitialDelay** () const
Returns the currently configured Max Inactivity Initial Delay duration.
- void **setMaxInactivityDurationInitialDelay** (long long maxInactivityDurationInitialDelay)
Sets the Max inactivity initial delay duration value.
- bool **isStackTraceEnabled** () const
Checks if the stackTraceEnabled flag is on.
- void **setStackTraceEnabled** (bool stackTraceEnabled)
Sets if the stackTraceEnabled flag is on.
- bool **isTcpNoDelayEnabled** () const
Checks if the tcpNoDelayEnabled flag is on.
- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)
Sets if the tcpNoDelayEnabled flag is on.
- bool **isCacheEnabled** () const
Checks if the cacheEnabled flag is on.
- void **setCacheEnabled** (bool cacheEnabled)
Sets if the cacheEnabled flag is on.
- int **getCacheSize** () const
Gets the Cache Size setting.
- void **setCacheSize** (int value)
Sets the Cache Size setting.
- bool **isTightEncodingEnabled** () const
Checks if the tightEncodingEnabled flag is on.
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)
Sets if the tightEncodingEnabled flag is on.
- bool **isSizePrefixDisabled** () const
Checks if the sizePrefixDisabled flag is on.
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)
Sets if the sizePrefixDisabled flag is on.
- const std::vector< unsigned char > &**getMagic** () const
Get the Magic field.
- void **setMagic** (const std::vector< unsigned char > &magic)
Sets the value of the magic field.

- `const std::vector< unsigned char > & getMarshaledProperties () const`
Get the marshalledProperties field.
- `void setMarshaledProperties (const std::vector< unsigned char > &marshalledProperties)`
Sets the value of the marshalledProperties field.
- `virtual const util::PrimitiveMap & getProperties () const`
*Gets the Properties for this **Command** (p. 1063).*
- `virtual util::PrimitiveMap & getProperties ()`
*Gets the Properties for this **Command** (p. 1063).*
- `virtual void setProperties (const util::PrimitiveMap &map)`
*Sets the Properties for this **Command** (p. 1063).*
- `bool isValid () const`
*Determines if we think this is a Valid **WireFormatInfo** (p. 3369) command.*
- `virtual bool isWireFormatInfo () const`
- `virtual void beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.
- `virtual void afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)`
Called after unmarshaling is started to cleanup the object being unmarshaled.

Static Public Attributes

- `static const unsigned char ID_WIREFORMATINFO = 1`

6.731.1 Constructor & Destructor Documentation

- 6.731.1.1 `activemq::commands::WireFormatInfo::WireFormatInfo ()`
- 6.731.1.2 `virtual activemq::commands::WireFormatInfo::~~WireFormatInfo ()`
[virtual]

6.731.2 Member Function Documentation

- 6.731.2.1 `virtual void activemq::commands::WireFormatInfo::afterUnmarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)` [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters:

wireFormat - the wireformat object to control unmarshaling

Reimplemented from `activemq::commands::BaseDataStructure` (p. 716).

6.731.2.2 `virtual void activemq::commands::WireFormatInfo::beforeMarshal (wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException) [virtual]`

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

Parameters:

wireFormat - the wire formatting controller

Reimplemented from `activemq::commands::BaseDataStructure` (p. 716).

6.731.2.3 `virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1461).

6.731.2.4 `virtual void activemq::commands::WireFormatInfo::copyDataStructure (const DataStructure *src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.731.2.5 `virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure *value) const [virtual]`

Compares the `DataStructure` (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 651).

6.731.2.6 `int activemq::commands::WireFormatInfo::getCacheSize () const`

Gets the Cache Size setting.

Returns:

currently set cache size.

6.731.2.7 `virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataSet** (p. 1461) type copy.

Implements **activemq::commands::DataSet** (p. 1464).

6.731.2.8 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic () const [inline]`

Get the Magic field.

Returns:

const reference to a `std::vector<char>`

6.731.2.9 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties () const [inline]`

Get the marshalledProperties field.

Returns:

const reference to a `std::vector<char>`

6.731.2.10 `long long activemq::commands::WireFormatInfo::getMaxInactivityDuration () const`

Returns the currently configured Max Inactivity duration.

Returns:

the set inactivity duration value.

6.731.2.11 `long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitialDelay () const`

Returns the currently configured Max Inactivity Initial Delay duration.

Returns:

the set inactivity duration initial delay value.

6.731.2.12 `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () [inline, virtual]`

Gets the Properties for this **Command** (p. 1063).

Returns:

the Properties object for this **Command** (p. 1063).

6.731.2.13 `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () const [inline, virtual]`

Gets the Properties for this **Command** (p. 1063).

Returns:

the Properties object for this **Command** (p. 1063).

6.731.2.14 `int activemq::commands::WireFormatInfo::getVersion () const [inline]`

Get the current Wireformat Version.

Returns:

int that identifies the version

6.731.2.15 `bool activemq::commands::WireFormatInfo::isCacheEnabled () const`

Checks if the cacheEnabled flag is on.

Returns:

true if the flag is on.

6.731.2.16 `virtual bool activemq::commands::WireFormatInfo::isMarshalAware () const [inline, virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns:

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 717).

6.731.2.17 **bool** **activemq::commands::WireFormatInfo::isSizePrefixDisabled** ()
 const

Checks if the sizePrefixDisabled flag is on.

Returns:

true if the flag is on.

6.731.2.18 **bool** **activemq::commands::WireFormatInfo::isStackTraceEnabled** ()
 const

Checks if the stackTraceEnabled flag is on.

Returns:

true if the flag is on.

6.731.2.19 **bool** **activemq::commands::WireFormatInfo::isTcpNoDelayEnabled** ()
 const

Checks if the tcpNoDelayEnabled flag is on.

Returns:

true if the flag is on.

6.731.2.20 **bool** **activemq::commands::WireFormatInfo::isTightEncodingEnabled** ()
 const

Checks if the tightEncodingEnabled flag is on.

Returns:

true if the flag is on.

6.731.2.21 **bool** **activemq::commands::WireFormatInfo::isValid** () **const**

Determines if we think this is a Valid **WireFormatInfo** (p. 3369) command.

Returns:

true if its valid.

6.731.2.22 `virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo ()
 const [inline, virtual]`

Returns:

answers true to the isWireFormatInfo query

Reimplemented from `activemq::commands::BaseCommand` (p. 655).

6.731.2.23 `void activemq::commands::WireFormatInfo::setCacheEnabled (bool
 cacheEnabled)`

Sets if the cacheEnabled flag is on.

Parameters:

cacheEnabled - true to turn flag is on

6.731.2.24 `void activemq::commands::WireFormatInfo::setCacheSize (int value)`

Sets the Cache Size setting.

Parameters:

value - value to set to the cache size.

6.731.2.25 `void activemq::commands::WireFormatInfo::setMagic (const
 std::vector< unsigned char > & magic) [inline]`

Sets the value of the magic field.

Parameters:

magic - const std::vector<char>

6.731.2.26 `void activemq::commands::WireFormatInfo::setMarshaledProperties
 (const std::vector< unsigned char > & marshalledProperties) [inline]`

Sets the value of the marshalledProperties field.

Parameters:

marshalledProperties The Byte Array vector that contains the marshaled form of the Message (p. 2145) properties, this is the data sent over the wire.

6.731.2.27 `void activemq::commands::WireFormatInfo::setMaxInactivityDuration
 (long long maxInactivityDuration)`

Sets the Max inactivity duration value.

Parameters:

maxInactivityDuration - max time a client can be inactive.

6.731.2.28 void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitialDelay (long long *maxInactivityDurationInitialDelay*)

Sets the Max inactivity initial delay duration value.

Parameters:

maxInactivityDurationInitialDelay - time before the inactivity delay is checked.

6.731.2.29 virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & *map*) [inline, virtual]

Sets the Properties for this **Command** (p. 1063).

Parameters:

map - PrimitiveMap to copy

6.731.2.30 void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool *sizePrefixDisabled*)

Sets if the sizePrefixDisabled flag is on.

Parameters:

sizePrefixDisabled - true to turn flag is on

6.731.2.31 void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool *stackTraceEnabled*)

Sets if the stackTraceEnabled flag is on.

Parameters:

stackTraceEnabled - ture to turn flag is on

6.731.2.32 void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*)

Sets if the tcpNoDelayEnabled flag is on.

Parameters:

tcpNoDelayEnabled - ture to turn flag is on

6.731.2.33 void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool *tightEncodingEnabled*)

Sets if the tightEncodingEnabled flag is on.

Parameters:

tightEncodingEnabled - true to turn flag is on

6.731.2.34 `void activemq::commands::WireFormatInfo::setVersion (int version)`
 [inline]

Set the current Wireformat Version.

Parameters:

version - int that identifies the version

6.731.2.35 `virtual std::string activemq::commands::WireFormatInfo::toString ()`
 const [virtual]

Returns a string containing the information for this **DataStructure** (p.1461) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 655).

6.731.2.36 `virtual decaf::lang::Pointer<commands::Command>`
 `activemq::commands::WireFormatInfo::visit (ac-`
 `tivemq::state::CommandVisitor * visitor) throw (`
 `exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2781) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.1067).

6.731.3 Field Documentation

6.731.3.1 `const unsigned char activemq::commands::WireFormatInfo::ID_-`
 `WIREFORMATINFO = 1 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**WireFormatInfo.h**

6.732 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3379).

#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.732.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3379). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.732.2 Constructor & Destructor Documentation

6.732.2.1 `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

6.732.2.2 `virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

6.732.3 Member Function Documentation

6.732.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.732.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.732.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.732.3.4 virtual void **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.732.3.5 virtual int **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.732.3.6 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.732.3.7 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**

6.733 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3383).

#include <src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.733.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3383). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.733.2 Constructor & Destructor Documentation

6.733.2.1 `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

6.733.2.2 `virtual activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

6.733.3 Member Function Documentation

6.733.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.733.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.733.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.733.3.4 virtual void **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.733.3.5 virtual int **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.733.3.6 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.733.3.7 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**WireFormatInfoMarshaller.h**

6.734 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3387).

#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.734.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3387). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.734.2 Constructor & Destructor Documentation

6.734.2.1 `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` `[inline]`

6.734.2.2 `virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` `[inline, virtual]`

6.734.3 Member Function Documentation

6.734.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.734.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.734.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1430).

6.734.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1436).

6.734.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1442).

6.734.3.6 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.734.3.7 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h**

6.735 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3391).

#include <src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.735.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3391). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.735.2 Constructor & Destructor Documentation

6.735.2.1 `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

6.735.2.2 `virtual activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

6.735.3 Member Function Documentation

6.735.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.735.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.735.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.735.3.4 virtual void **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.735.3.5 virtual int **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.735.3.6 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.735.3.7 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**WireFormatInfoMarshaller.h**

6.736 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3395).

#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h>Inheritance diagram for activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.736.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3395). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.736.2 Constructor & Destructor Documentation

6.736.2.1 `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` `[inline]`

6.736.2.2 `virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` `[inline, virtual]`

6.736.3 Member Function Documentation

6.736.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.736.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.736.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1430).

6.736.3.4 virtual void **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1436).

6.736.3.5 virtual int **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1442).

6.736.3.6 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1448).

6.736.3.7 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1454).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**WireFormatInfoMarshaller.h**

6.737 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 3399) which allows a **WireFormat** (p. 3363) to.

#include <src/main/activemq/wireformat/WireFormatNegotiator.h>Inheritance diagram for activemq::wireformat::WireFormatNegotiator:

Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)
Constructor.
- virtual ~**WireFormatNegotiator** ()

6.737.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 3399) which allows a **WireFormat** (p. 3363) to.

6.737.2 Constructor & Destructor Documentation

6.737.2.1 activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator (const **Pointer**< **transport::Transport** > & *next*) [inline]

Constructor.

Parameters:

next - the next Transport in the chain

6.737.2.2 virtual activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

6.738 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 3363) Factories that are available to the client at runtime.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

Public Member Functions

- virtual **~WireFormatRegistry** ()
- **WireFormatFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)

*Gets a Registered **WireFormatFactory** (p. 3367) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **WireFormatFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

*Registers a new **WireFormatFactory** (p. 3367) with this Registry.*

- void **unregisterFactory** (const std::string &name)

Unregisters the Factory with the given name and deletes that instance of the Factory.

- std::vector< std::string > **getWireFormatNames** () const

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()

*Gets the single instance of the **WireFormatRegistry** (p. 3400).*

6.738.1 Detailed Description

Registry of all **WireFormat** (p. 3363) Factories that are available to the client at runtime. New WireFormat's must have a factory registered here before a connection attempt is made.

Since:

3.0

6.738.2 Constructor & Destructor Documentation

6.738.2.1 `virtual`
`activemq::wireformat::WireFormatRegistry::~~WireFormatRegistry ()`
`[virtual]`

6.738.3 Member Function Documentation

6.738.3.1 `WireFormatFactory* ac-`
`tivemq::wireformat::WireFormatRegistry::findFactory (const std::string`
`& name) const throw (decaf::lang::exceptions::NoSuchElementException`
`)`

Gets a Registered **WireFormatFactory** (p. 3367) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters:

name The name of the Factory to find in the Registry.

Returns:

the Factory registered under the given name.

Exceptions:

NoSuchElementException if no factory is registered with that name.

6.738.3.2 `static WireFormatRegistry& ac-`
`tivemq::wireformat::WireFormatRegistry::getInstance ()`
`[static]`

Gets the single instance of the **WireFormatRegistry** (p. 3400).

Returns:

reference to the single instance of this Registry

6.738.3.3 `std::vector<std::string> ac-`
`tivemq::wireformat::WireFormatRegistry::getWireFormatNames ()`
`const`

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Returns:

stl vector of strings with all the **WireFormat** (p. 3363) names registered.

6.738.3.4 void activemq::wireformat::WireFormatRegistry::registerFactory
(const std::string & *name*, WireFormatFactory * *factory*)
throw (decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::NullPointerException)

Registers a new **WireFormatFactory** (p. 3367) with this Registry. If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters:

name The name of the new Factory to register.
factory The new Factory to add to the Registry.

Exceptions:

IllegalArgumentException is name is the empty string.
NullPointerException if the Factory is Null.

6.738.3.5 void activemq::wireformat::WireFormatRegistry::unregisterFactory
(const std::string & *name*)

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters:

name Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatRegistry.h**

6.739 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {.

#include <src/main/activemq/transport/inactivity/WriteChecker.h>Inheritance diagram for activemq::transport::inactivity::WriteChecker:

Public Member Functions

- **WriteChecker** (**InactivityMonitor** *parent)
- virtual ~**WriteChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.739.1 Detailed Description

Runnable class used by the {.

See also:

InactivityMonitor (p. 1733)} to make periodic writes to the underlying **transport** (p. 79) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

Since:

3.1.0

6.739.2 Constructor & Destructor Documentation

6.739.2.1 **activemq::transport::inactivity::WriteChecker::WriteChecker** (**InactivityMonitor** * *parent*)

6.739.2.2 **virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker** ()
[virtual]

6.739.3 Member Function Documentation

6.739.3.1 **virtual void activemq::transport::inactivity::WriteChecker::run** ()
[virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2816).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**WriteChecker.h**

6.740 decaf::io::Writer Class Reference

```
#include <src/main/decaf/io/Writer.h>
```

Public Member Functions

- virtual `~Writer ()`
- virtual void `setOutputStream (OutputStream *os)=0`
Sets the target output stream.
- virtual `OutputStream * getOutputStream ()=0`
Gets the target output stream.
- virtual void `write (const unsigned char *buffer, std::size_t count)=0 throw (IOException, lang::exceptions::NullPointerException)`
Writes a byte array to the output stream.
- virtual void `writeByte (unsigned char v)=0 throw (IOException)`
Writes a byte to the output stream.

6.740.1 Constructor & Destructor Documentation

6.740.1.1 virtual `decaf::io::Writer::~~Writer ()` [inline, virtual]

6.740.2 Member Function Documentation

6.740.2.1 virtual `OutputStream* decaf::io::Writer::getOutputStream ()` [pure virtual]

Gets the target output stream.

Returns:

the output stream currently being used

6.740.2.2 virtual void `decaf::io::Writer::setOutputStream (OutputStream * os)` [pure virtual]

Sets the target output stream.

Parameters:

os The provided OutputStream to use to write to.

6.740.2.3 virtual void `decaf::io::Writer::write (const unsigned char * buffer, std::size_t count) throw (IOException, lang::exceptions::NullPointerException)` [pure virtual]

Writes a byte array to the output stream.

Parameters:

buffer a byte array

count the number of bytes in the array to write.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

6.740.2.4 `virtual void decaf::io::Writer::writeByte (unsigned char v) throw (IOException) [pure virtual]`

Writes a byte to the output stream.

Parameters:

v The value to be written.

Exceptions:

IOException (p. 1820) thrown if an error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Writer.h`

6.741 decaf::security::auth::x500::X500Principal Class Reference

#include <src/main/decaf/security/auth/x500/X500Principal.h> Inheritance diagram for decaf::security::auth::x500::X500Principal:

Public Member Functions

- virtual `~X500Principal ()`
- virtual `std::string getName () const =0`
Provides the name of this principal.
- virtual `void getEncoded (std::vector< unsigned char > &output) const =0`
- virtual `int hashCode () const =0`

6.741.1 Constructor & Destructor Documentation

6.741.1.1 virtual `decaf::security::auth::x500::X500Principal::~X500Principal ()`
[inline, virtual]

6.741.2 Member Function Documentation

6.741.2.1 virtual `void decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & output) const` [pure virtual]

6.741.2.2 virtual `std::string decaf::security::auth::x500::X500Principal::getName () const` [pure virtual]

Provides the name of this principal.

Returns:

the name of this principal.

Implements `decaf::security::Principal` (p. 2569).

6.741.2.3 virtual `int decaf::security::auth::x500::X500Principal::hashCode () const`
[pure virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/auth/x500/X500Principal.h`

6.742 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

#include <src/main/decaf/security/cert/X509Certificate.h> Inheritance diagram for decaf::security::cert::X509Certificate:

Public Member Functions

- virtual `~X509Certificate ()`
- virtual void `checkValidity ()` const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void `checkValidity (const decaf::util::Date &date)` const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int `getBasicConstraints ()` const =0
- virtual void `getIssuerUniqueID (std::vector< bool > &output)` const =0
- virtual const X500Principal * `getIssuerX500Principal ()` const =0
- virtual void `getKeyUsage (std::vector< unsigned char > &output)` const =0
- virtual Date `getNotAfter ()` const =0
- virtual Date `getNotBefore ()` const =0
- virtual std::string `getSigAlgName ()` const =0
- virtual std::string `getSigAlgOID ()` const =0
- virtual void `getSigAlgParams (std::vector< unsigned char > &output)` const =0
- virtual void `getSignature (std::vector< unsigned char > &output)` const =0
- virtual void `getSubjectUniqueID (std::vector< bool > &output)` const =0
- virtual const X500Principal * `getSubjectX500Principal ()` const =0
- virtual void `getTBSCertificate (std::vector< unsigned char > &output)` const =0 throw (CertificateEncodingException)
- virtual int `getVersion ()` const =0

6.742.1 Detailed Description

Base interface for all identity certificates.

6.742.2 Constructor & Destructor Documentation

- 6.742.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate ()
[inline, virtual]

6.742.3 Member Function Documentation

- 6.742.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implemented in decaf::security_provider::unix::openssl::OpenSSLX509Certificate (p. 2443).

6.742.3.2 `virtual void decaf::security::cert::X509Certificate::checkValidity () const
throw (CertificateExpiredException, CertificateNotYetValidException)
[pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2443).

6.742.3.3 `virtual int decaf::security::cert::X509Certificate::getBasicConstraints ()
const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2444).

6.742.3.4 `virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID
(std::vector< bool > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2444).

6.742.3.5 `virtual const X500Principal* de-
caf::security::cert::X509Certificate::getIssuerX500Principal
() const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2444).

6.742.3.6 `virtual void decaf::security::cert::X509Certificate::getKeyUsage
(std::vector< unsigned char > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2444).

6.742.3.7 `virtual Date decaf::security::cert::X509Certificate::getNotAfter () const
[pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2444).

6.742.3.8 `virtual Date decaf::security::cert::X509Certificate::getNotBefore () const
[pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2444).

6.742.3.9 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgName
() const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2445).

6.742.3.10 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID
() const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2445).

6.742.3.11 `virtual void decaf::security::cert::X509Certificate::getSigAlgParams
(std::vector< unsigned char > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2445).

6.742.3.12 `virtual void decaf::security::cert::X509Certificate::getSignature
(std::vector< unsigned char > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2445).

6.742.3.13 `virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID
(std::vector< bool > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2446).

6.742.3.14 `virtual const X500Principal* de-
caf::security::cert::X509Certificate::getSubjectX500Principal () const
[pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2446).

6.742.3.15 `virtual void decaf::security::cert::X509Certificate::getTBSCertificate
(std::vector< unsigned char > & output) const throw (
CertificateEncodingException) [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2446).

6.742.3.16 `virtual int decaf::security::cert::X509Certificate::getVersion () const
[pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate`
(p. 2446).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/X509Certificate.h`

6.743 activemq::commands::XATransactionId Class Reference

#include <src/main/activemq/commands/XATransactionId.h> Inheritance diagram for activemq::commands::XATransactionId:

Public Types

- typedef decaf::lang::PointerComparator< XATransactionId > COMPARATOR

Public Member Functions

- XATransactionId ()
- XATransactionId (const XATransactionId &other)
- virtual ~XATransactionId ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual XATransactionId * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const DataStructure *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
Returns a string containing the information for this DataStructure (p. 1461) such as its type and value of its elements.
- virtual bool **equals** (const DataStructure *value) const
Compares the DataStructure (p. 1461) passed in to this one, and returns if they are equivalent.
- virtual int **getFormatId** () const
- virtual void **setFormatId** (int formatId)
- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const
- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()
- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &globalTransactionId)
- virtual const std::vector< unsigned char > & **getBranchQualifier** () const
- virtual std::vector< unsigned char > & **getBranchQualifier** ()
- virtual void **setBranchQualifier** (const std::vector< unsigned char > &branchQualifier)
- virtual int **compareTo** (const XATransactionId &value) const
- virtual bool **equals** (const XATransactionId &value) const
- virtual bool **operator==** (const XATransactionId &value) const
- virtual bool **operator<** (const XATransactionId &value) const
- XATransactionId & **operator=** (const XATransactionId &other)

Static Public Attributes

- static const unsigned char **ID_XATRANSACTIONID** = 112

Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

6.743.1 Member Typedef Documentation

- 6.743.1.1** `typedef decaf::lang::PointerComparator<XATransactionId>
activemq::commands::XATransactionId::COMPARATOR`

Reimplemented from `activemq::commands::TransactionId` (p. 3217).

6.743.2 Constructor & Destructor Documentation

- 6.743.2.1** `activemq::commands::XATransactionId::XATransactionId ()`
- 6.743.2.2** `activemq::commands::XATransactionId::XATransactionId (const
XATransactionId & other)`
- 6.743.2.3** `virtual activemq::commands::XATransactionId::~~XATransactionId ()
[virtual]`

6.743.3 Member Function Documentation

- 6.743.3.1** `virtual XATransactionId* ac-
tivemq::commands::XATransactionId::cloneDataStructure
() const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from `activemq::commands::TransactionId` (p. 3218).

- 6.743.3.2** `virtual int activemq::commands::XATransactionId::compareTo (const
XATransactionId & value) const [virtual]`
- 6.743.3.3** `virtual void activemq::commands::XATransactionId::copyDataStructure
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from `activemq::commands::TransactionId` (p. 3218).

6.743.3.4 `virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & value) const` [virtual]

6.743.3.5 `virtual bool activemq::commands::XATransactionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1461) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::TransactionId` (p. 3218).

6.743.3.6 `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ()` [virtual]

6.743.3.7 `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () const` [virtual]

6.743.3.8 `virtual unsigned char activemq::commands::XATransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1461) type copy.

Reimplemented from `activemq::commands::TransactionId` (p. 3219).

- | | |
|------------|---|
| 6.743.3.9 | virtual int activemq::commands::XATransactionId::getFormatId () const
[virtual] |
| 6.743.3.10 | virtual std::vector<unsigned char>& ac-
tivemq::commands::XATransactionId::getGlobalTransactionId ()
[virtual] |
| 6.743.3.11 | virtual const std::vector<unsigned char>& ac-
tivemq::commands::XATransactionId::getGlobalTransactionId () const
[virtual] |
| 6.743.3.12 | virtual bool activemq::commands::XATransactionId::operator< (const
XATransactionId & <i>value</i>) const [virtual] |
| 6.743.3.13 | XATransactionId& activemq::commands::XATransactionId::operator=
(const XATransactionId & <i>other</i>) |
| 6.743.3.14 | virtual bool activemq::commands::XATransactionId::operator== (const
XATransactionId & <i>value</i>) const [virtual] |
| 6.743.3.15 | virtual void activemq::commands::XATransactionId::setBranchQualifier
(const std::vector< unsigned char > & <i>branchQualifier</i>) [virtual] |
| 6.743.3.16 | virtual void activemq::commands::XATransactionId::setFormatId (int
<i>formatId</i>) [virtual] |
| 6.743.3.17 | virtual void ac-
tivemq::commands::XATransactionId::setGlobalTransactionId (const
std::vector< unsigned char > & <i>globalTransactionId</i>) [virtual] |
| 6.743.3.18 | virtual std::string activemq::commands::XATransactionId::toString ()
const [virtual] |

6.743.4 Field Documentation

- 6.743.4.1 `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier` [protected]
- 6.743.4.2 `int activemq::commands::XATransactionId::formatId` [protected]
- 6.743.4.3 `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId` [protected]
- 6.743.4.4 `const unsigned char activemq::commands::XATransactionId::ID_XATRANSACTIONID = 112` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

6.744 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3415).

#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.744.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3415). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.744.2 Constructor & Destructor Documentation

6.744.2.1 `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransactionIdMarshaller()` `[inline]`

6.744.2.2 `virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` `[inline, virtual]`

6.744.3 Member Function Documentation

6.744.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.744.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.744.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3237).

6.744.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3237).

6.744.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3238).

6.744.3.6 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3238).

6.744.3.7 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3239).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h

6.745 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3419).

#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.745.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3419). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.745.2 Constructor & Destructor Documentation

6.745.2.1 `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransactionIdMarshaller()` `[inline]`

6.745.2.2 `virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` `[inline, virtual]`

6.745.3 Member Function Documentation

6.745.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.745.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.745.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3221).

6.745.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3221).

6.745.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3222).

6.745.3.6 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3222).

6.745.3.7 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3223).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h

6.746 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3423).

#include <src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.746.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3423). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.746.2 Constructor & Destructor Documentation

6.746.2.1 `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::XATransactionIdMarshaller()` `[inline]`

6.746.2.2 `virtual activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` `[inline, virtual]`

6.746.3 Member Function Documentation

6.746.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.746.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.746.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3229).

6.746.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3229).

6.746.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3230).

6.746.3.6 virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3230).

6.746.3.7 virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3231).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h

6.747 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3427).

#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.747.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3427). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.747.2 Constructor & Destructor Documentation

6.747.2.1 `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransactionIdMarshaller()` [inline]

6.747.2.2 `virtual activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` [inline, virtual]

6.747.3 Member Function Documentation

6.747.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.747.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.747.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3225).

6.747.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3225).

6.747.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3226).

6.747.3.6 virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3226).

6.747.3.7 virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3227).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h

6.748 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3431).

#include <src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h> Inheritance diagram for activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.748.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3431). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.748.2 Constructor & Destructor Documentation

6.748.2.1 `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::XATransactionIdMarshaller()` `[inline]`

6.748.2.2 `virtual activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::~~XATransactionIdMarshaller()` `[inline, virtual]`

6.748.3 Member Function Documentation

6.748.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1419).

6.748.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1424).

6.748.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3233).

6.748.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3233).

6.748.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3234).

6.748.3.6 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3234).

6.748.3.7 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3235).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h

Chapter 7

File Documentation

7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedConsumer**
A cached message consumer contained within a pooled session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedProducer**
A cached message producer contained within a pooled session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::cmsutil::CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 1041) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1184) to operate on.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```

Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**
Extends the `CmsAccessor` (p. 1024) to add support for resolving destination names.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

Data Structures

- class **activemq::cmsutil::CmsTemplate**
CmsTemplate (p. 1041) simplifies performing synchronous CMS operations.
- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DestinationResolver**
*Resolves a CMS destination name to a *Destination*.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**
*Resolves a CMS destination name to a *Destination*.*
- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**
Manages maps of names to topics and queues for a single session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::MessageCreator**
Creates the user-defined message to be sent by the `CmsTemplate` (p. 1041).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/STLMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::PooledSession**
A pooled session object that wraps around a delegate session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ProducerCallback**
Callback for sending a message to a CMS destination.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**
Manages the lifecycle of a set of CMS resources.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.12 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionCallback**
Callback for executing any number of operations on a provided CMS Session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.13 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionPool**
A pool of CMS sessions from the same connection and with the same acknowledge mode.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.14 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQBlobMessage`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.15 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBytesMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.16 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.17 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQMapMessage`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.18 src/main/activemq/commands/ActiveMQMessage.h File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessage**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.19 src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <activemq/util/CMSExceptionSupport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class `activemq::commands::ActiveMQMessageTemplate< T >`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.20 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQObjectMessage`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.21 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQQueue`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.22 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQStreamMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.23 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

Data Structures

- class **activemq::commands::ActiveMQTempDestination**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**
- namespace **activemq::commands**

7.24 src/main/activemq/commands/ActiveMQTempQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQTempQueue`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.25 src/main/activemq/commands/ActiveMQTempTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQTempTopic`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.26 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQTextMessage`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.27 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQTopic`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.28 src/main/activemq/commands/BaseCommand.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
```

Data Structures

- class **activemq::commands::BaseCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.29 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <string>
#include <sstream>
```

Data Structures

- class **activemq::commands::BaseDataStructure**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::commands**

7.30 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::commands::BooleanExpression**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.31 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerError**
This class represents an Exception sent from the Broker.
- struct **activemq::commands::BrokerError::StackTraceElement**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.32 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::BrokerId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.33 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.34 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::commands::Command**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**
- namespace **activemq::commands**

7.35 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionControl`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.36 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionError**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.37 src/main/activemq/commands/ConnectionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.38 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.39 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConsumerControl`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.40 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.41 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.42 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ControlCommand`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.43 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DataArrayResponse**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.44 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DataResponse`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.45 src/main/activemq/commands/DataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

Data Structures

- class **activemq::commands::DataStructure**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.46 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DestinationInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.47 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DiscoveryEvent`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.48 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ExceptionResponse`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.49 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::FlushCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.50 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::IntegerResponse`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.51 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalQueueAck**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.52 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::JournalTopicAck`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.53 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTrace**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.54 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTransaction**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.55 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::KeepAliveInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.56 src/main/activemq/commands/LastPartialCommand.h File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::LastPartialCommand`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.57 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::LocalTransactionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.58 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataSet.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Message**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.59 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <cms/CMSException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::Message**
Root of all messages.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.60 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageAck**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.61 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageDispatch`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.62 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageDispatchNotification**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.63 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.64 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessagePull`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.65 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::NetworkBridgeFilter`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.66 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::PartialCommand`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.67 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerAck**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.68 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.69 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ProducerInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.70 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataSet.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::RemoveInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.71 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.72 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ReplayCommand`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.73 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Response**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.74 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionId**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.75 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.76 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ShutdownInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.77 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SubscriptionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.78 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::TransactionId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.79 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::TransactionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.80 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

Data Structures

- class **activemq::commands::WireFormatInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.81 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::XATransactionId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.82 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQAckHandler**
Interface class that is used to give CMS Messages an interface to Ack themselves with.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**
- namespace **activemq::core**

7.83 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionSupport.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/util/Properties.h>
#include <decaf/util/STLMap.h>
#include <decaf/util/STLSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnection**
Concrete connection used for all connectors to the ActiveMQ broker.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.84 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.85 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 233) class.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.86 src/main/activemq/core/ActiveMQConnectionSupport.h File Reference

```
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnectionSupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQConstants**
Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.
- class **activemq::core::ActiveMQConstants::StaticInitializer**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConsumer**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.89 src/main/activemq/core/ActiveMQProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQProducer**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.90 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQSession**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.91 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::ActiveMQSessionExecutor**
Delegate dispatcher for a single session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.92 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::core::ActiveMQTransactionContext**
Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.93 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::DispatchData**

Simple POCO that contains the information necessary to route a message to a specified consumer.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.94 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::Dispatcher**

Interface for an object responsible for dispatching messages to consumers.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.95 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::core::MessageDispatchChannel`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::core`

7.96 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::core::Synchronization**
*Transacted Object **Synchronization** (p. 3137), used to sync the events of a Transaction with the items in the Transaction.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.97 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <decaf/lang/Exception.h>
#include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class `activemq::exceptions::ActiveMQException`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::exceptions`

7.98 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

Data Structures

- class `activemq::exceptions::BrokerException`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::exceptions`

7.99 src/main/activemq/exceptions/ExceptionDefines.h File Reference

Defines

- `#define AMQ_CATCH_RETHROW(type)`
Macro for catching and re-throwing an exception of a given type.
- `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`
Macro for catching an exception of one type and then re-throwing as another type.
- `#define AMQ_CATCHALL_THROW(type)`
A catch-all that throws a known exception.
- `#define AMQ_CATCHALL_NOTHROW()`
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.
- `#define AMQ_CATCH_NOTHROW(type)`
Macro for catching and re-throwing an exception of a given type.

7.99.1 Define Documentation

7.99.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

Parameters:

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.99.1.2 `#define AMQ_CATCH_NOTHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. `ActiveMQException`).

7.99.1.3 `#define AMQ_CATCH_RETHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. `ActiveMQException`).

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.99.1.4 `#define AMQ_CATCHALL_NOTHROW()`

Value:

```
catch( ... ){ \
    activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
    "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

7.99.1.5 `#define AMQ_CATCHALL_THROW(type)`

Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
    "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters:

type the type of exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`,
`activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`,
and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.100 src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference

Defines

- `#define DECAF_CATCH_RETHROW(type)`
Macro for catching and rethrowing an exception of a given type.
- `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`
Macro for catching an exception of one type and then rethrowing as another type.
- `#define DECAF_CATCHALL_THROW(type)`
A catch-all that throws a known exception.
- `#define DECAF_CATCHALL_NOTHROW()`
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.
- `#define DECAF_CATCH_NOTHROW(type)`
Macro for catching and rethrowing an exception of a given type.

7.100.1 Define Documentation

7.100.1.1 `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

Parameters:

sourceType the type of the exception to be caught.
targetType the type of the exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.

7.100.1.2 `#define DECAF_CATCH_NOTHROW(type)`

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. `Exception`).

Referenced by `decaf::io::FilterInputStream::~~FilterInputStream()`, `decaf::io::FilterOutputStream::~~FilterOutputStream()`, and `decaf::util::logging::StreamHandler::~~StreamHandler()`.

7.100.1.3 #define DECAF_CATCH_RETHROW(type)**Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. `Exception`).

Referenced by `decaf::util::PriorityQueue< E >::add()`, `decaf::io::FilterInputStream::available()`, `decaf::io::FilterInputStream::close()`, `decaf::io::FilterInputStream::close()`, `decaf::io::FilterOutputStream::flush()`, `decaf::util::concurrent::Lock::lock()`, `decaf::util::concurrent::Lock::lock()`, `decaf::util::logging::StreamHandler::publish()`, `decaf::io::FilterInputStream::read()`, `decaf::io::FilterInputStream::reset()`, `decaf::io::FilterInputStream::skip()`, `decaf::util::concurrent::Lock::unlock()`, `decaf::io::FilterOutputStream::write()`, and `decaf::util::concurrent::Lock::~~Lock()`.

7.100.1.4 #define DECAF_CATCHALL_NOTHROW()**Value:**

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

Referenced by `decaf::io::FilterInputStream::mark()`, `decaf::io::FilterInputStream::markSupported()`, `decaf::io::FilterInputStream::~~FilterInputStream()`, `decaf::io::FilterOutputStream::~~FilterOutputStream()`, and `decaf::util::logging::StreamHandler::~~StreamHandler()`.

7.100.1.5 #define DECAF_CATCHALL_THROW(type)**Value:**

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters:

type the type of exception to be thrown.

Referenced by decaf::util::PriorityQueue< E >::add(), decaf::io::FilterInputStream::available(), decaf::io::FilterOutputStream::close(), decaf::io::FilterInputStream::close(), decaf::io::FilterOutputStream::flush(), decaf::util::concurrent::Lock::lock(), decaf::util::concurrent::Lock::Lock(), decaf::util::logging::StreamHandler::publish(), decaf::io::FilterInputStream::read(), decaf::io::FilterInputStream::reset(), decaf::io::FilterInputStream::skip(), decaf::util::concurrent::Lock::unlock(), decaf::io::FilterOutputStream::write(), and decaf::util::concurrent::Lock::~~Lock().

7.101 src/main/activemq/io/LoggingInputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class `activemq::io::LoggingInputStream`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::io`

7.102 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

Data Structures

- class **activemq::io::LoggingOutputStream**
OutputStream filter that just logs the data being written.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::io**

7.103 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class `activemq::library::ActiveMQCPP`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::library`

7.104 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::CommandVisitor**

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**
- namespace **activemq::state**

7.105 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

Data Structures

- class **activemq::state::CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1069) that returns NULL for all calls.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.106 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::ConnectionState`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.107 src/main/activemq/state/ConnectionStateTracker.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::state::ConnectionStateTracker`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.108 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::ConsumerState`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.109 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::ProducerState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.110 src/main/activemq/state/SessionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::SessionState`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.111 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::state::Tracked`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.112 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::TransactionState`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.113 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::CompositeTask**
Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 1092).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.114 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::CompositeTaskRunner**

*A **Task** (p. 3148) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.115 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::threads::DedicatedTaskRunner`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::threads`

7.116 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::threads::Task**

Represents a unit of work that requires one or more iterations to complete.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.117 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::TaskRunner**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.118 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::AbstractTransportFactory**
*Abstract implementation of the **TransportFactory** (p. 3279) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3279) instances.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.119 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::transport::CompositeTransport**

*A Composite **Transport** (p. 3273) is a **Transport** (p. 3273) implementation that is composed of several **Transports**.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.120 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::correlator::FutureResponse**

A container that holds a response object.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.121 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.122 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::DefaultTransportListener**

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::transport**

7.123 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class `activemq::transport::failover::BackupTransport`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::transport`
- namespace `activemq::transport::failover`

7.124 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

Data Structures

- class `activemq::transport::failover::BackupTransportPool`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::transport`
- namespace `activemq::transport::failover`

7.125 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::transport::failover::CloseTransportsTask`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::transport`
- namespace `activemq::transport::failover`

7.126 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.127 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**
*Creates an instance of a **FailoverTransport** (p. 1612).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.128 src/main/activemq/transport/failover/FailoverTransportListener. File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportListener**
*Utility class used by the **Transport** (p. 3273) to perform the work of responding to events from the active **Transport** (p. 3273).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.129 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

Data Structures

- class **activemq::transport::failover::URIPool**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.130 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Timer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Data Structures

- class **activemq::transport::inactivity::InactivityMonitor**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.131 src/main/activemq/transport/inactivity/ReadChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

Data Structures

- class **activemq::transport::inactivity::ReadChecker**
Runnable class that is used by the {}.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.132 src/main/activemq/transport/inactivity/WriteChecker.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/util/TimerTask.h>
```

Data Structures

- class **activemq::transport::inactivity::WriteChecker**

Runnable class used by the {}.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.133 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

Data Structures

- class **activemq::transport::IOTransport**

*Implementation of the **Transport** (p. 3273) interface that performs marshaling of commands to IO streams.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.134 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::logging::LoggingTransport**
A transport filter that logs commands as they are sent/received.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::logging**

7.135 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

Data Structures

- class **activemq::transport::mock::InternalCommandListener**
*Listens for Commands sent from the **MockTransport** (p. 2371).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.136 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

Data Structures

- class **activemq::transport::mock::MockTransport**

*The **MockTransport** (p. 2371) defines a base level **Transport** (p. 3273) class that is intended to be used in place of an a regular protocol **Transport** (p. 3273) such as TCP.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.137 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```

Data Structures

- class **activemq::transport::mock::MockTransportFactory**
Manufactures MockTransports, which are objects that read from input streams and write to output streams.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.138 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

Data Structures

- class **activemq::transport::mock::ResponseBuilder**
Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.139 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

Data Structures

- class **activemq::transport::tcp::TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1823).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.140 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**
*Factory Responsible for creating the **TcpTransport** (p. 3160).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.141 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::Transport**
Interface for a transport layer for command objects.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::transport**

7.142 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportFactory**
Defines the interface for Factories that create Transports or TransportFilters.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.143 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::TransportFilter**

A filter on the transport layer.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.144 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportListener**
A listener of asynchronous exceptions from a command transport object.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.145 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::transport::TransportRegistry**
*Registry of all **Transport** (p. 3273) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.146 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::util::ActiveMQProperties**
*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2657) object.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.147 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <cms/CMSSecurityException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/InvalidClientIdException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/InvalidSelectorException.h>
#include <cms/IllegalStateException.h>
#include <cms/UnsupportedOperationException.h>
#include <decaf/lang/Exception.h>
#include <string>
```

Data Structures

- class **activemq::util::CMSExceptionSupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

Defines

- **#define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()**
Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

7.147.1 Define Documentation

7.147.1.1 #define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need

to throw an exception of MessageNotReadableException for instance.

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
 >::acknowledge(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
 >::clearBody(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
 >::clearProperties(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
 >::equals(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
 >::getBooleanProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getByteProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getCMSMessageID(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getIntProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getLongProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getPropertyNames(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getShortProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::getStringProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::propertyExists(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setByteProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setCMSDestination(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setCMSReplyTo(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setIntProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setLongProperty(), activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setShortProperty(), and activemq::commands::ActiveMQMessageTemplate<
 cms::ObjectMessage >::setStringProperty().

7.148 src/main/activemq/util/CompositeData.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>
```

Data Structures

- class **activemq::util::CompositeData**
Represents a Composite URI.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.149 src/main/activemq/util/Config.h File Reference

Defines

- `#define AMQCPP_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`

7.149.1 Define Documentation

7.149.1.1 `#define AMQCPP_API`

7.149.1.2 `#define HAVE_PTHREAD_H`

7.149.1.3 `#define HAVE_UUID_T`

7.149.1.4 `#define HAVE_UUID_UUID_H`

7.150 src/main/cms/Config.h File Reference

Defines

- `#define CMS_API`

7.150.1 Define Documentation

7.150.1.1 `#define CMS_API`

7.151 src/main/decaf/util/Config.h File Reference

Defines

- `#define DECAF_API`
- `#define DECAF_UNUSED`

7.151.1 Define Documentation

7.151.1.1 `#define DECAF_API`

7.151.1.2 `#define DECAF_UNUSED`

7.152 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::LongSequenceGenerator**

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.153 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::MemoryUsage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.154 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveList**
List of primitives.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.155 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveMap**
Map of named primitives.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.156 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

Data Structures

- class **activemq::util::PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2555) from one type to another.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.157 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::util::PrimitiveValueNode**
Class that wraps around a single value of one of the many types.
- union **activemq::util::PrimitiveValueNode::PrimitiveValue**
Define a union type comprised of the various types.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.158 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::util::URISupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.159 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::Usage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.160 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::MarshalAware**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.161

src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h

File Reference

7.161 src/main/activemq/wireformat/openwire/marshal/BaseDataStrea³⁶⁰⁷

File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

```
#include <activemq/wireformat/openwire/utils/HexTable.h>
```

```
#include <activemq/commands/MessageId.h>
```

```
#include <activemq/commands/ProducerId.h>
```

```
#include <activemq/commands/TransactionId.h>
```

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**
Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.162 src/main/activemq/wireformat/openwire/marshal/DataStreamM File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq:wireformat::openwire::marshal::DataStreamMarshaller**
Base class for all classes that marshal commands for Openwire.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq:wireformat**
- namespace **activemq:wireformat::openwire**
- namespace **activemq:wireformat::openwire::marshal**

7.163

src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File

Reference

7.163 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h 3609

File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.164 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 181).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.165 src/main/activemq/wireformat/openwire/marsh
shal/v2/ActiveMQBlobMessageMarshaller.h File

Reference

7.165 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶¹¹/v2/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 193).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.166 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 177).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.167 src/main/activemq/wireformat/openwire/marsh
shal/v4/ActiveMQBlobMessageMarshaller.h File

Reference

7.167 ³⁶¹³src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 185).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.168 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.169 src/main/activemq/wireformat/openwire/marsh-
shal/v1/ActiveMQBytesMessageMarshaller.h File

Reference

~~7.169~~ ³⁶¹⁵ src/main/activemq/wireformat/openwire/marsh/v1/ActiveMQ
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 217).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.170 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 229).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.171 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQBytesMessageMarshaller.h File

Reference

7.171 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶¹⁷~~shal/v3/ActiveMQ~~
File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 213).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.172 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 221).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.173 src/main/activemq/wireformat/openwire/marsh
shal/v5/ActiveMQBytesMessageMarshaller.h File

Reference

7.173 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶¹⁹/ActiveMQ
File Reference

```
#include <activemq/wireformat/openwire/marshall/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 225).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.174 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 289).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.175 src/main/activemq/wireformat/openwire/marsh
shal/v2/ActiveMQDestinationMarshaller.h File

Reference

7.175 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶²¹/ActiveMQ
File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 301).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.176 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 285).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.177 src/main/activemq/wireformat/openwire/marsh
shal/v4/ActiveMQDestinationMarshaller.h File

Reference

7.177 ³⁶²³src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 293).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.178 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 297).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.179 src/main/activemq/wireformat/openwire/marsh
shal/v1/ActiveMQMapMessageMarshaller.h File

Reference

~~7.179~~ src/main/activemq/wireformat/openwire/marsh³⁶²⁵al/v1/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 326).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.180 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 338).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.181 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQMapMessageMarshaller.h File

Reference

~~7.181~~ src/main/activemq/wireformat/openwire/marsh³⁶²⁷shal/v3/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 322).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.182 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 330).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.183 src/main/activemq/wireformat/openwire/marsh
shal/v5/ActiveMQMapMessageMarshaller.h File

Reference

7.183 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶²⁹/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 334).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.184 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 349).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.185 src/main/activemq/wireformat/openwire/mar-
shal/v2/ActiveMQMessageMarshaller.h File

Reference

7.185 ³⁶³¹src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshall/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshall::v2::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 361).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshall**
- namespace **activemq::wireformat::openwire::marshall::v2**

7.186 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 345).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.187 src/main/activemq/wireformat/openwire/mar-
shal/v4/ActiveMQMessageMarshaller.h File

Reference

~~7.187~~ ³⁶³³ src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 353).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.188 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 357).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.189 src/main/activemq/wireformat/openwire/marsh-
shal/v1/ActiveMQObjectMessageMarshaller.h File

Reference

~~7.189~~ ³⁶³⁵ src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQO
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 390).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.190 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 402).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.191 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQObjectMessageMarshaller.h File

Reference

~~7.191~~ ³⁶³⁷src/main/activemq/wireformat/openwire/marsh/v3/ActiveMQ
File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 386).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.192 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 394).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.193 src/main/activemq/wireformat/openwire/marsh
shal/v5/ActiveMQObjectMessageMarshaller.h File

Reference

7.193 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶³⁹/ActiveMQ
File Reference

```
#include <activemq/wireformat/openwire/marshall/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 398).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.194 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 427).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.195

src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h

File Reference

7.195 ³⁶⁴¹src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**
Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 439).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.196 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 423).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.197

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

File Reference

7.197 ³⁶⁴³src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 431).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.198 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 435).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.199 src/main/activemq/wireformat/openwire/marsh
shal/v1/ActiveMQStreamMessageMarshaller.h File

Reference

~~7.199~~ ³⁶⁴⁵ src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQS
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 483).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.200 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 495).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.201 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQStreamMessageMarshaller.h File

Reference

7.201 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶⁴⁷/ActiveMQS

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 479).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.202 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 487).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.203 src/main/activemq/wireformat/openwire/marsh
shal/v5/ActiveMQStreamMessageMarshaller.h File

Reference

7.203 ³⁶⁴⁹src/main/activemq/wireformat/openwire/marsh/v5/ActiveMQS

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 491).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.204 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 507).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.205 src/main/activemq/wireformat/openwire/marsh
shal/v2/ActiveMQTempDestinationMarshaller.h File

Reference

7.205 ³⁶⁵¹src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 519).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.206 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 503).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.207 src/main/activemq/wireformat/openwire/marsh
shal/v4/ActiveMQTempDestinationMarshaller.h File

Reference

7.207 ³⁶⁵³src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 511).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.208 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 515).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.209 src/main/activemq/wireformat/openwire/marsh
shal/v1/ActiveMQTempQueueMarshaller.h File

Reference

7.209 ³⁶⁵⁵src/main/activemq/wireformat/openwire/marsh/v1/ActiveMQ
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 532).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.210 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 544).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.211 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQTempQueueMarshaller.h File

Reference

~~7.211~~ ³⁶⁵⁷src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 528).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.212 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 536).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.213 src/main/activemq/wireformat/openwire/marsh
shal/v5/ActiveMQTempQueueMarshaller.h File

Reference

7.213 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶⁵⁹/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 540).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.214 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 557).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.215 src/main/activemq/wireformat/openwire/marsh
shal/v2/ActiveMQTempTopicMarshaller.h File

Reference

7.215 ³⁶⁶¹src/main/activemq/wireformat/openwire/marsh/v2/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 569).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.216 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 553).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.217 src/main/activemq/wireformat/openwire/marsh
shal/v4/ActiveMQTempTopicMarshaller.h File

Reference

7.217 ³⁶⁶³src/main/activemq/wireformat/openwire/marsh/v4/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 561).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.218 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 565).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.219 src/main/activemq/wireformat/openwire/marsh
shal/v1/ActiveMQTextMessageMarshaller.h File

Reference

7.219 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶⁶⁵~~shal/v1/ActiveMQ~~

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 582).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.220 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 594).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.221 src/main/activemq/wireformat/openwire/marsh
shal/v3/ActiveMQTextMessageMarshaller.h File

Reference

7.221 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶⁶⁷/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 578).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.222 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 586).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.223 src/main/activemq/wireformat/openwire/marsh
shal/v5/ActiveMQTextMessageMarshaller.h File

Reference

7.223 ~~src/main/activemq/wireformat/openwire/marsh~~³⁶⁶⁹/ActiveMQ

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 590).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.224 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 606).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.225

src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h

File Reference

7.225 ³⁶⁷¹ src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 618).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.226 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 602).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.227

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h

File Reference

7.227 ³⁶⁷³ src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 610).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.228 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 614).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.229

src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h

File Reference

7.229 src/main/activemq/wireformat/openwire/marshal/v1/BaseComm³⁶⁷⁵

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 664).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.230 src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 685).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.231

src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h

File Reference

7.231 ³⁶⁷⁷ src/main/activemq/wireformat/openwire/marshal/v3/BaseComm

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 657).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.232 src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 671).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.233

src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h

File Reference

7.233 src/main/activemq/wireformat/openwire/marshal/v5/BaseComm³⁶⁷⁹

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 678).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.234 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 759).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.235 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 771).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.236 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 755).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.237 src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 763).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.238 src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 767).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.239 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 787).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.240 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 799).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.241 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 783).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.242 src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 791).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.243 src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 795).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.244 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1144).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.245 src/main/activemq/wireformat/openwire/marsh
shal/v2/ConnectionControlMarshaller.h File

Reference

7.245 ³⁶⁹¹src/main/activemq/wireformat/openwire/marsh/v2/Connection

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1156).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.246 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1140).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.247 src/main/activemq/wireformat/openwire/marsh
shal/v4/ConnectionControlMarshaller.h File

Reference

7.247 ³⁶⁹³src/main/activemq/wireformat/openwire/marshal/v4/Connection

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1148).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.248 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1152).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.249

src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h

File Reference

7.249 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h 3695

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1168).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.250 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1180).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.251

src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

File Reference

7.251 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h 3697

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1164).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.252 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1172).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.253

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h

File Reference

7.253 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h 3699

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1176).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.254 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1195).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.255

src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h

File Reference

7.255 src/main/activemq/wireformat/openwire/marshal/v2/Connection³⁷⁰¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1207).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.256 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1191).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.257

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

File Reference

7.257 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h 3703

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1199).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.258 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1203).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.259

src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h

File Reference

7.259 src/main/activemq/wireformat/openwire/marshal/v1/Connection

3705

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1225).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.260 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1233).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.261

src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h

File Reference

7.261 ³⁷⁰⁷src/main/activemq/wireformat/openwire/marshal/v3/Connection

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1217).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.262 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1221).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.263

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h

File Reference

7.263 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h 3709

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1229).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.264 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1263).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.265 src/main/activemq/wireformat/openwire/marsh
shal/v2/ConsumerControlMarshaller.h File

Reference

7.265 src/main/activemq/wireformat/openwire/marsh³⁷¹¹/v2/ConsumerC

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1271).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.266 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1255).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.267 src/main/activemq/wireformat/openwire/marsh
shal/v4/ConsumerControlMarshaller.h File

Reference

7.267 src/main/activemq/wireformat/openwire/marsh³⁷¹³
shal/v4/ConsumerC

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1259).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.268 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1267).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.269

src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File

Reference

7.269 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h 3715

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1288).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.270 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1296).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.271

src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File

Reference

7.271 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1280).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.272 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1284).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1292).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.274 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1313).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.275

src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

File Reference

7.275 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h³⁷²¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1325).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.276 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1309).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.277

src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

File Reference

7.277 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h 3723

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1317).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.278 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1321).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.279 src/main/activemq/wireformat/openwire/marsh
shal/v1/ControlCommandMarshaller.h File

Reference

7.279 ³⁷²⁵src/main/activemq/wireformat/openwire/marshal/v1/ControlCo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1342).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.280 src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1350).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.281 src/main/activemq/wireformat/openwire/marsh
shal/v3/ControlCommandMarshaller.h File

Reference

7.281 ~~src/main/activemq/wireformat/openwire/marsh~~³⁷²⁷/v3/ControlCo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1334).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.282 src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1338).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.283 src/main/activemq/wireformat/openwire/mar-
shal/v5/ControlCommandMarshaller.h File

Reference

7.283 ~~src/main/activemq/wireformat/openwire/marsh~~³⁷²⁹/v5/ControlCo

File Reference

```
#include <activemq/wireformat/openwire/marshall/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1346).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.284 src/main/activemq/wireformat/openwire/marshal/v1/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1367).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.285 src/main/activemq/wireformat/openwire/marsh
shal/v2/DataArrayResponseMarshaller.h File

Reference

7.285 ³⁷³¹src/main/activemq/wireformat/openwire/marsh/v2/DataArray
File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1375).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.286 src/main/activemq/wireformat/openwire/marshal/v3/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1359).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.287 src/main/activemq/wireformat/openwire/marsh
shal/v4/DataArrayResponseMarshaller.h File

Reference

7.287 ³⁷³³src/main/activemq/wireformat/openwire/marsh/v4/DataArray
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1363).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.288 src/main/activemq/wireformat/openwire/marshal/v5/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1371).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.289

src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h

File Reference

7.289 src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h 3735

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1402).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.290 src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1414).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.291

src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h

File Reference

7.291 ³⁷³⁷src/main/activemq/wireformat/openwire/marshal/v3/DataResponse

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1398).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.292 src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1410).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.293

src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h

File Reference

7.293 src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h 3739

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1406).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.294 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1501).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.295

src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h

File Reference

7.295 ³⁷⁴¹src/main/activemq/wireformat/openwire/marshal/v2/Destination

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1489).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.296 src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1493).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.297

src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h

File Reference

7.297 src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h 3743

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1497).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.298 src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1505).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.299

src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h

File Reference

7.299 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryE³⁷⁴⁵

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1531).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.300 src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1515).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.301

src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h

File Reference

7.301 ³⁷⁴⁷src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryE

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1519).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-ments.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.302 src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1523).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.303

src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

File Reference

7.303 src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h 3749

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1527).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.304 src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1589).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.305 src/main/activemq/wireformat/openwire/marsh
shal/v2/ExceptionResponseMarshaller.h File

Reference

7.305 src/main/activemq/wireformat/openwire/marsh³⁷⁵¹
shal/v2/ExceptionResponseMarshaller.h File

File Reference

```
#include <activemq/wireformat/openwire/marsh/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1585).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.306 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1593).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.307 src/main/activemq/wireformat/openwire/marsh
shal/v4/ExceptionResponseMarshaller.h File

Reference

7.307 src/main/activemq/wireformat/openwire/marsh³⁷⁵³
shal/v4/ExceptionResponseMarshaller.h File

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1597).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.308 src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1601).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.309

src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h

File Reference

7.309 src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h 3755

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1683).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.310 src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1679).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.311

src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h

File Reference

7.311 src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h 3757

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1687).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.312 src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1691).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.313

src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h

File Reference

7.313 src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h 3759

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1695).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.314 src/main/activemq/wireformat/openwire/marshal/v1/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1784).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.315

src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h

File Reference

7.315 src/main/activemq/wireformat/openwire/marshal/v2/IntegerRes³⁷⁶¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1780).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.316 src/main/activemq/wireformat/openwire/marshal/v3/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1788).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.317

src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h

File Reference

7.317 src/main/activemq/wireformat/openwire/marshal/v4/IntegerRes³⁷⁶³

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1792).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.318 src/main/activemq/wireformat/openwire/marshal/v5/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1796).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.319 src/main/activemq/wireformat/openwire/mar-
shal/v1/JournalQueueAckMarshaller.h File

Reference

7.319 ~~src/main/activemq/wireformat/openwire/marsh~~³⁷⁶⁵~~al/v1/JournalQu~~
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1850).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.320 src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1838).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.321 src/main/activemq/wireformat/openwire/marsh-
shal/v3/JournalQueueAckMarshaller.h File

Reference

7.321 ³⁷⁶⁷src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1846).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.322 src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1842).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.323 src/main/activemq/wireformat/openwire/marsh-
shal/v5/JournalQueueAckMarshaller.h File

Reference

7.323 ³⁷⁶⁹src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1854).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.324 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1867).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.325 src/main/activemq/wireformat/openwire/marsh-
shal/v2/JournalTopicAckMarshaller.h File

Reference

7.325 src/main/activemq/wireformat/openwire/marsh³⁷⁷¹/v2/JournalTop

File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1863).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.326 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1871).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.327 src/main/activemq/wireformat/openwire/marsh-
shal/v4/JournalTopicAckMarshaller.h File

Reference

7.327 src/main/activemq/wireformat/openwire/marsh³⁷⁷³/v4/JournalTop

File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1875).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.328 src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1879).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.329

src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h

File Reference

7.329 ³⁷⁷⁵ src/main/activemq/wireformat/openwire/marshal/v1/JournalTrace

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1898).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.330 src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1886).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.331

src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h

File Reference

~~7.331~~ ³⁷⁷⁷ src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1894).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.332 src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1890).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.333

src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h

File Reference

~~7.333~~ src/main/activemq/wireformat/openwire/marshal/v5/JournalTra³⁷⁷⁹

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1902).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.334 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1922).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.335 src/main/activemq/wireformat/openwire/marsh-
shal/v2/JournalTransactionMarshaller.h File

Reference

7.335 ³⁷⁸¹src/main/activemq/wireformat/openwire/marshal/v2/JournalTra

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1910).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.336 src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1914).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.337 src/main/activemq/wireformat/openwire/marsh
shal/v4/JournalTransactionMarshaller.h File

Reference

7.337 ³⁷⁸³src/main/activemq/wireformat/openwire/marshal/v4/JournalTra
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1918).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.338 src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1926).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.339

src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

File Reference

7.339 ³⁷⁸⁵ src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1949).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.340 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1933).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.341

src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

File Reference

7.341 ³⁷⁸⁷src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1941).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.342 src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1945).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.343

src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h

File Reference

7.343 ³⁷⁸⁹ src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1937).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.344 src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1965).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.345 src/main/activemq/wireformat/openwire/marsh
shal/v2/LastPartialCommandMarshaller.h File

Reference

7.345 ³⁷⁹¹src/main/activemq/wireformat/openwire/marsh/v2/LastPartial

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1961).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.346 src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1969).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.347 src/main/activemq/wireformat/openwire/marsh
shal/v4/LastPartialCommandMarshaller.h File

Reference

7.347 src/main/activemq/wireformat/openwire/marsh³⁷⁹³shal/v4/LastPartial

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1973).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.348 src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1977).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.349 src/main/activemq/wireformat/openwire/marsh
shal/v1/LocalTransactionIdMarshaller.h File

Reference

7.349 ~~src/main/activemq/wireformat/openwire/marsh~~³⁷⁹⁵/v1/LocalTrans
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2009).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.350 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1997).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.351 src/main/activemq/wireformat/openwire/marsh
shal/v3/LocalTransactionIdMarshaller.h File

Reference

7.351 src/main/activemq/wireformat/openwire/marsh³⁷⁹⁷/v3/LocalTrans

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2001).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.352 src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2005).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.353 src/main/activemq/wireformat/openwire/marsh
shal/v5/LocalTransactionIdMarshaller.h File

Reference

7.353 src/main/activemq/wireformat/openwire/marsh³⁷⁹⁹/v5/LocalTrans

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2013).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.354 src/main/activemq/wireformat/openwire/marshal/v1/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.355 src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.356 src/main/activemq/wireformat/openwire/marshal/v3/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.357 src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.358 src/main/activemq/wireformat/openwire/marshal/v5/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.359

src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h File

Reference

7.359 src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h 3805

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2210).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.360 src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2194).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.361

src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h File

Reference

7.361 src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h 3807

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2202).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.362 src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2206).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.363

src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h File

Reference

7.363 src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h 3809

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2198).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.364 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2243).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.365 src/main/activemq/wireformat/openwire/mar-
shal/v2/MessageDispatchMarshaller.h File

Reference

7.365 ³⁸¹¹src/main/activemq/wireformat/openwire/marsh/v2/MessageDi
File Reference

```
#include <activemq/wireformat/openwire/marshall/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2231).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.366 src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2239).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.367 src/main/activemq/wireformat/openwire/marsh
shal/v4/MessageDispatchMarshaller.h File

Reference

7.367 src/main/activemq/wireformat/openwire/marsh³⁸¹³/v4/MessageDi
File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2235).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.368 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshall.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2247).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.369 src/main/activemq/wireformat/openwire/marsh
shal/v1/MessageDispatchNotificationMarshaller.h File

Reference

7.369 src/main/activemq/wireformat/openwire/marsh³⁸¹⁵al/v1/MessageDi
File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 2269).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.370 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2257).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.371 src/main/activemq/wireformat/openwire/mar-
shal/v3/MessageDispatchNotificationMarshaller.h File

Reference

7.371 src/main/activemq/wireformat/openwire/marsh³⁸¹⁷/v3/MessageDi
File Reference

```
#include <activemq/wireformat/openwire/marshall/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
(p. 2265).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.372 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2261).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.373 src/main/activemq/wireformat/openwire/marsh
shal/v5/MessageDispatchNotificationMarshaller.h File

Reference

7.373 src/main/activemq/wireformat/openwire/marsh³⁸¹⁹/v5/MessageDi
File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller
(p. 2273).

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.374 src/main/activemq/wireformat/openwire/marshal/v1/MessageId. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2300).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.375 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2284).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.376 src/main/activemq/wireformat/openwire/marshal/v3/MessageId. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2292).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.377 src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2288).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.378 src/main/activemq/wireformat/openwire/marshal/v5/MessageId. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2296).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.379 src/main/activemq/wireformat/openwire/marshal/v1/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2325).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.380 src/main/activemq/wireformat/openwire/marshal/v2/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2305).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.381 src/main/activemq/wireformat/openwire/marshal/v3/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2315).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.382 src/main/activemq/wireformat/openwire/marshal/v4/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2310).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.383 src/main/activemq/wireformat/openwire/marshal/v5/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2320).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.384 src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2359).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.385

src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

File Reference

7.385 src/main/activemq/wireformat/openwire/marshal/v2/MessagePu³⁸³¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2351).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.386 src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2355).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.387

src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

File Reference

7.387 src/main/activemq/wireformat/openwire/marshal/v4/MessagePu³⁸³³

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2367).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.388 src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2363).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.389 src/main/activemq/wireformat/openwire/marsh
shal/v1/NetworkBridgeFilterMarshaller.h File

Reference

7.389 ³⁸³⁵src/main/activemq/wireformat/openwire/marshal/v1/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2409).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.390 src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2397).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.391 src/main/activemq/wireformat/openwire/marsh
shal/v3/NetworkBridgeFilterMarshaller.h File

Reference

7.391 ~~src/main/activemq/wireformat/openwire/marsh~~³⁸³⁷/v3/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2401).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.392 src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2413).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.393 src/main/activemq/wireformat/openwire/marsh
shal/v5/NetworkBridgeFilterMarshaller.h File

Reference

7.393 src/main/activemq/wireformat/openwire/marsh³⁸³⁹/v5/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2405).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.394 src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2489).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.395 src/main/activemq/wireformat/openwire/marsh-
shal/v2/PartialCommandMarshaller.h File

Reference

7.395 ³⁸⁴¹src/main/activemq/wireformat/openwire/marshal/v2/PartialCon
File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2477).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.396 src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2481).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.397 src/main/activemq/wireformat/openwire/marsh-
shal/v4/PartialCommandMarshaller.h File

Reference

7.397 ³⁸⁴³src/main/activemq/wireformat/openwire/marshal/v4/PartialCon

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2485).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.398 src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2493).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.399

src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h

File Reference

7.399 src/main/activemq/wireformat/openwire/marshal/v1/ProducerA³⁸⁴⁵

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2599).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.400 src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2583).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.401

src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h

File Reference

7.401 ³⁸⁴⁷src/main/activemq/wireformat/openwire/marshal/v3/ProducerA

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2587).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.402 src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2591).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.403

src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h

File Reference

7.403 src/main/activemq/wireformat/openwire/marshal/v5/ProducerA³⁸⁴⁹

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2595).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.404 src/main/activemq/wireformat/openwire/marshal/v1/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2623).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.405

src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h File

Reference

7.405 ³⁸⁵¹src/main/activemq/wireformat/openwire/marshal/v2/ProducerId

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2611).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.406 src/main/activemq/wireformat/openwire/marshal/v3/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2615).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.407

src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h File

Reference

7.407 ³⁸⁵³src/main/activemq/wireformat/openwire/marshal/v4/ProducerId

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2627).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.408 src/main/activemq/wireformat/openwire/marshal/v5/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2619).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.409

src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h

File Reference

7.409 ³⁸⁵⁵ src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2648).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.410 src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2636).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.411

src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h

File Reference

7.411 ³⁸⁵⁷src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2652).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.412 src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2644).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.413

src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h

File Reference

7.413 ³⁸⁵⁹ src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2640).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.414 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2707).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.415

src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h File

Reference

7.415 ³⁸⁶¹src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2711).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.416 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2715).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.417

src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h File

Reference

7.417 ³⁸⁶³src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2719).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.418 src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2723).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.419 src/main/activemq/wireformat/openwire/marsh
shal/v1/RemoveSubscriptionInfoMarshaller.h File

Reference

7.419 ~~src/main/activemq/wireformat/openwire/marsh~~³⁸⁶⁵~~al/v1/RemoveSu~~

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2740).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.420 src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2736).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.421 src/main/activemq/wireformat/openwire/marsh
shal/v3/RemoveSubscriptionInfoMarshaller.h File

Reference

7.421 ~~src/main/activemq/wireformat/openwire/marsh~~³⁸⁶⁷~~shal/v3/RemoveSu~~

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2744).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.422 src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2748).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.423 src/main/activemq/wireformat/openwire/marsh
shal/v5/RemoveSubscriptionInfoMarshaller.h File

Reference

7.423 src/main/activemq/wireformat/openwire/marsh³⁸⁶⁹/v5/RemoveSub

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2732).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.424 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2772).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.425 src/main/activemq/wireformat/openwire/marsh-
shal/v2/ReplayCommandMarshaller.h File

Reference

7.425 ³⁸⁷¹src/main/activemq/wireformat/openwire/marshal/v2/ReplayCon

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2756).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.426 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2760).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.427 src/main/activemq/wireformat/openwire/mar-
shal/v4/ReplayCommandMarshaller.h File

Reference

7.427 ³⁸⁷³src/main/activemq/wireformat/openwire/marsh/v4/ReplayCon

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2768).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.428 src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2764).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.429 src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2806).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.430 src/main/activemq/wireformat/openwire/marshal/v2/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2791).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.431 src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2796).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.432 src/main/activemq/wireformat/openwire/marshal/v4/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2811).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.433 src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2801).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.434 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2861).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.435 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2865).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.436 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2869).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.437 src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2873).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.438 src/main/activemq/wireformat/openwire/marshal/v5/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2857).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.439

src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h File

Reference

7.439 ³⁸⁸⁵src/main/activemq/wireformat/openwire/marshal/v1/SessionInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2881).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.440 src/main/activemq/wireformat/openwire/marshal/v2/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2889).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.441

src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h File

Reference

7.441³⁸⁸⁷ src/main/activemq/wireformat/openwire/marshal/v3/SessionInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2897).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.442 src/main/activemq/wireformat/openwire/marshal/v4/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2885).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.443

src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h File

Reference

7.443³⁸⁸⁹ src/main/activemq/wireformat/openwire/marshal/v5/SessionInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2893).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.444 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2937).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.445

src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h

File Reference

7.445 ³⁸⁹¹ src/main/activemq/wireformat/openwire/marshal/v2/ShutdownI

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2949).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.446 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2945).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.447

src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h

File Reference

7.447 ³⁸⁹³src/main/activemq/wireformat/openwire/marshal/v4/ShutdownI

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2941).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.448 src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2953).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.449

src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

File Reference

7.449 ³⁸⁹⁵ src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3102).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.450 src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3118).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.451

src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h

File Reference

7.451 ³⁸⁹⁷src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3110).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.452 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3106).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.453

src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h

File Reference

7.453 src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h 3899

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3114).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.454 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3220).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.455

src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h

File Reference

7.455 src/main/activemq/wireformat/openwire/marshal/v2/Transaction³⁹⁰¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3224).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.456 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3236).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.457

src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h

File Reference

7.457 src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h 3903

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3228).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.458 src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3232).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.459

src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h

File Reference

7.459 src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h 3905

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3245).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.460 src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3261).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.461

src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h

File Reference

7.461 ³⁹⁰⁷src/main/activemq/wireformat/openwire/marshal/v3/Transaction

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3253).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.462 src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3249).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.463

src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h

File Reference

7.463 src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h 3909

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3257).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.464 src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3395).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.465

src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h

File Reference

7.465 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h 3911

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3387).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.466 src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3379).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.467

src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

File Reference

7.467 src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h 3913

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3391).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.468 src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3383).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.469 src/main/activemq/wireformat/openwire/marsh-
shal/v1/XATransactionIdMarshaller.h File

Reference

7.469 src/main/activemq/wireformat/openwire/marsh³⁹¹⁵/v1/XATransac

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3419).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.470 src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3427).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.471 src/main/activemq/wireformat/openwire/marsh
shal/v3/XATransactionIdMarshaller.h File

Reference

7.471 src/main/activemq/wireformat/openwire/marsh³⁹¹⁷/v3/XATransac

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3415).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.472 src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3423).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.473 src/main/activemq/wireformat/openwire/marsh
shal/v5/XATransactionIdMarshaller.h File

Reference

7.473 src/main/activemq/wireformat/openwire/marsh³⁹¹⁹/v5/XATransac

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3431).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.474 src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.475 src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.476 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq:wireformat::openwire::OpenWireFormatNegotiator**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq:wireformat**
- namespace **activemq:wireformat::openwire**

7.477 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.478 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq:wireformat::openwire::utils::BooleanStream**
Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq:wireformat**
- namespace **activemq:wireformat::openwire**
- namespace **activemq:wireformat::openwire::utils**

7.479 src/main/activemq/wireformat/openwire/Utils/HexTable.h File Reference

```
#include <vector>
#include <string>
#include <activemq/Util/Config.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **activemq::wireformat::openwire::Utils::HexTable**

*The **HexTable** (p. 1715) class maps hexadecimal strings to the value of an index into the table, i.e.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::Utils**

7.480 src/main/activemq/wireformat/openwire/utils/MessagePropertyL File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::MessagePropertyInterceptor**
Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.481 src/main/activemq/wireformat/openwire/Utils/OpenwireStringSupport.h File Reference

```
#include <string>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/Util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::Utils::OpenwireStringSupport**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::Utils**

7.482 src/main/activemq/wireformat/stomp/StompCommandConstants File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.483 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompFrame**
A Stomp-level message frame that encloses all messages to and from the broker.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.484 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompHelper**
Utility Methods used when marshaling to and from StompFrame's.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.485 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.486 src/main/activemq/wireformat/stomp/StompWireFormatFactory File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**
Factory used to create the Stomp Wire Format instance.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.487 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **activemq::wireformat::WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.488 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatFactory**

*The **WireFormatFactory** (p. 3367) is the interface that all **WireFormatFactory** (p. 3367) classes must extend.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.489 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::WireFormatNegotiator**
*Defines a **WireFormatNegotiator** (p. 3399) which allows a **WireFormat** (p. 3363) to.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.490 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatRegistry**
*Registry of all **WireFormat** (p. 3363) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.491 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

Data Structures

- class **cms::BytesMessage**

*A **BytesMessage** (p. 938) object is used to send a message containing a stream of unsigned bytes.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.492 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Closeable**
Interface for a class that implements the close method.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.493 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::Closeable**
Interface for a class that implements the close method.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.494 src/main/cms/CMSException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

Data Structures

- class **cms::CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.495 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

Data Structures

- class **cms::CMSProperties**
Interface for a Java-like properties object.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.496 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::CMSSecurityException**

This exception must be thrown when a provider rejects a user name/password submitted by a client.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.497 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stopable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **cms::Connection**
The client's connection to its provider.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.498 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1131) objects returned implement the CMS **Connection** (p. 1131) interface and hide the CMS Provider specific implementation details behind that interface.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.499 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ConnectionMetaData**

*A **ConnectionMetaData** (p. 1237) object provides information describing the **Connection** (p. 1131) object.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.500 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.501 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

Data Structures

- class **cms::Destination**

*A **Destination** (p. 1480) object encapsulates a provider-specific address.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.502 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1581) that is registered with the **Connection** (p. 1131).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.503 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::IllegalStateException**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.504 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.505 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidClientIdException**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.506 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidDestinationException**

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.507 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidSelectorException**

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.508 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::MapMessage**

*A **MapMessage** (p. 2106) object is used to send a set of name-value pairs.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.509 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

Data Structures

- class **cms::MessageConsumer**

A client uses a `MessageConsumer` (p. 2214) to received messages from a destination.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.510 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3081) or **BytesMessage** (p. 938) is being read.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.511 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageFormatException**

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.512 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::MessageListener**

A `MessageListener` (p. 2304) object is used to receive asynchronously delivered messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.513 src/main/cms/MessageNotReadableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotReadableException**

This exception must be thrown when a CMS client attempts to read a write-only message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.514 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotWriteableException**

This exception must be thrown when a CMS client attempts to write to a read-only message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.515 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
#include <cms/CMSException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/MessageFormatException.h>
#include <cms/UnsupportedOperationException.h>
#include <cms/DeliveryMode.h>
```

Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 2332) object to send messages to a **Destination** (p. 1480).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.516 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
```

Data Structures

- class **cms::ObjectMessage**

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.517 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Queue**

An interface encapsulating a provider-specific queue name.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.518 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::util::Queue< E >**
A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.519 src/main/cms/QueueBrowser.h File Reference

```
#include <vector>
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::QueueBrowser**
*This class implements in interface for browsing the messages in a **Queue** (p. 2674) without removing them.*

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.520 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Session**

*A **Session** (p. 2839) object is a single-threaded context for producing and consuming messages.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.521 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Startable**
Interface for a class that implements the start method.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.522 src/main/cms/Stoppable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Stoppable**
Interface for a class that implements the stop method.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.523 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

Data Structures

- class **cms::StreamMessage**
*Interface for a **StreamMessage** (p. 3081).*

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.524 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryQueue**
*Defines a Temporary **Queue** (p. 2674) based **Destination** (p. 1480).*

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.525 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryTopic**
*Defines a Temporary **Topic** (p. 3215) based **Destination** (p. 1480).*

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.526 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::TextMessage**
Interface for a text message.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.527 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Topic**

An interface encapsulating a provider-specific topic name.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.528 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::UnsupportedOperationException**

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.529 src/main/decaf/lang/exceptions/UnsupportedOperationException File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.530 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::AprPool**

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**

7.531 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <memory>
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::DecafRuntime**
Handles APR initialization and termination.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**

7.532 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**
Wrapper Around the Standard error Output facility on the current platform.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.533 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardInputStream**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.534 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardOutputStream**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.535 src/main/decaf/internal/net/URLEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

Data Structures

- class **decaf::internal::net::URLEncoderDecoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.536 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

Data Structures

- class **decaf::internal::net::URIHelper**
Helper class used by the URI classes in encoding and decoding of URI's.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.537 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::net::URIType**
Basic type object that holds data that composes a given URI.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.538 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
```

Data Structures

- class **decaf::internal::nio::BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 128) package to create the various default version of the NIO interfaces.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.539 src/main/decaf/internal/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
```

Data Structures

- class **decaf::internal::nio::ByteBuffer**

This class defines six categories of operations upon byte buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.540 src/main/decaf/internal/nio/ByteArrayPerspective.h File Reference

```
#include <decaf/internal/util/ByteArrayAdapter.h>
```

Data Structures

- class **decaf::internal::nio::ByteArrayPerspective**

This class extends `ByteArray` to create a reference counted byte array that can be held and used by several different `ByteBuffers` and allow them to know on destruction whose job it is to delete the perspective.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.541 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.542 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.543 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.544 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.545 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.546 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.547 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

Data Structures

- class **decaf::internal::util::ByteArrayAdapter**
This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.
- union **decaf::internal::util::ByteArrayAdapter::Array**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.548 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::ConditionImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.549 src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::MutexImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.550 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.551 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/util/concurrent/TimeoutException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::Transferer< E >**
Shared internal API for dual stacks and queues.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.552 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/locks/LockSupport.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferQueue< E >**

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.553 src/main/decaf/internal/util/concurrent/TransferStack.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferStack< E >**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.554 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <pthread.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.555 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.556 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <pthread.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.557 src/main/decaf/internal/util/concurrent/windows/MutexHandle. File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.558 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::internal::util::HexStringParser**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.559 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::util::TimerTaskHeap**

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.560 src/main/decaf/io/BlockingByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::BlockingByteArrayInputStream**
This is a blocking version of a byte buffer stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.561 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.562 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.563 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
#include <algorithm>
```

Data Structures

- class **decaf::io::ByteArrayInputStream**

*Simple implementation of **InputStream** (p. 1740) that wraps around an STL Vector `std::vector<unsigned char>`.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.564 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::ByteArrayOutputStream**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.565 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.566 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```

Data Structures

- class **decaf::io::DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.567 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::EOFException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.568 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::FilterInputStream**

*A **FilterInputStream** (p. 1631) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.569 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::FilterOutputStream**

This class is the superclass of all classes that filter output streams.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.570 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::InputStream**
Base interface for an input stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.571 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::InterruptedIOException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.572 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::io::IOException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.573 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::OutputStream**
Base interface for an output stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.574 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Reader**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.575 src/main/decaf/io/UnsupportedEncodingException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UnsupportedEncodingException**
Thrown when the the Character Encoding is not supported.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.576 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IIOException.h>
```

Data Structures

- class **decaf::io::UTFDataFormatException**

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.577 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Writer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.578 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Appendable**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.579 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Boolean**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.580 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Byte**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.581 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::lang::Character**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.582 src/main/decaf/lang/CharSequence.h File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::CharSequence**

*A **CharSequence** (p. 1014) is a readable sequence of char values.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.583 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Comparable**< **T** >

This interface imposes a total ordering on the objects of each class that implements it.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.584 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Double**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.585 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class **decaf::lang::Exception**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.586 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::ClassCastException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.587 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.588 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.589 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalThreadStateException**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.590 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.591 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InterruptedException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.592 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.593 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.594 src/main/decaf/lang/exceptions/NullPointerException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NullPointerException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.595 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NumberFormatException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.596 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::RuntimeException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.597 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Float**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.598 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

Data Structures

- class **decaf::lang::Integer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.599 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/util/Iterator.h>
```

Data Structures

- class **decaf::lang::Iterable**< **E** >

*Implementing this interface allows an object to be cast to an **Iterable** (p. 1830) type for generic collections API calls.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.600 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Long**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.601 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Math**

*The class **Math** (p. 2126) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.602 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Number**

*The abstract class **Number** (p. 2432) is the superclass of classes **Byte** (p. 840), **Double** (p. 1537), **Float** (p. 1647), **Integer** (p. 1762), **Long** (p. 2057), and **Short** (p. 2906).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.603 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

Data Structures

- class **decaf::lang::AtomicRefCounter**
- struct **decaf::lang::STATIC_CAST_TOKEN**
- struct **decaf::lang::DYNAMIC_CAST_TOKEN**
- class **decaf::lang::Pointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 2497) that is a template on a Type and is Thread Safe if the default Reference Counter is used.*

- class **decaf::lang::PointerComparator< T, R >**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2497) instance.*

- struct **std::less< decaf::lang::Pointer< T > >**

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **std**

Functions

- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const Pointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator==** (const U *left, const Pointer< T, R > &right)
- template<typename T , typename R , typename U >
bool **decaf::lang::operator!=** (const Pointer< T, R > &left, const U *right)

- `template<typename T , typename R , typename U >`
 `bool decaf::lang::operator!= (const U *left, const Pointer< T, R > &right)`

7.604 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.605 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runtime**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.606 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Short**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.607 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/Exception.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

Data Structures

- class **decaf::lang::System**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.608 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Config.h>
```

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

7.609 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::ThreadGroup**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.610 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Throwable**
This class represents an error that has occurred.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.611 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::BindException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.612 src/main/decaf/net/BufferedSocket.h File Reference

```
#include <decaf/net/Socket.h>
#include <decaf/net/SocketException.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
```

Data Structures

- class **decaf::net::BufferedSocket**

*Buffered **Socket** (p. 2964) class that wraps a **Socket** (p. 2964) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.613 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::ConnectException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.614 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::HttpRetryException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.615 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::MalformedURLException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.616 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::NoRouteToHostException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.617 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::PortUnreachableException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.618 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::ProtocolException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.619 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/net/TcpSocket.h>
#include <decaf/net/SocketException.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
```

Data Structures

- class **decaf::net::ServerSocket**
A server socket class (for testing purposes).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.620 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <apr_network_io.h>
```

Data Structures

- class **decaf::net::Socket**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.621 src/main/decaf/net/SocketError.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketError**
Static utility class to simplify handling of error codes for socket operations.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.622 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::SocketException**
Exception for errors when manipulating sockets.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.623 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/util/Properties.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketFactory**
Socket (p. 2964) Factory implementation for use in Creating Sockets.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.624 src/main/decaf/net/SocketInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/net/Socket.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::net::SocketInputStream**
Input stream for performing reads on a socket.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.625 src/main/decaf/net/SocketOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/net/Socket.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::net::SocketOutputStream**
Output stream for performing write operations on a socket.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.626 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InterruptedIOException.h>
```

Data Structures

- class **decaf::net::SocketTimeoutException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.627 src/main/decaf/net/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/Socket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
```

Data Structures

- class **decaf::net::TcpSocket**
Platform-independent implementation of the socket interface.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.628 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownHostException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.629 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownServiceException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.630 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

Data Structures

- class **decaf::net::URI**

*This class represents an instance of a **URI** (p. 3308) as defined by RFC 2396.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.631 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::net::URISyntaxException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.632 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URL**

*Class **URL** (p. 3347) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.633 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

Data Structures

- class **decaf::net::URLDecoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.634 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

Data Structures

- class **decaf::net::URLEncoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.635 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/InvalidMarkException.h>
```

Data Structures

- class **decaf::nio::Buffer**
A container for data of a specific primitive type.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.636 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferOverflowException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.637 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferUnderflowException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.638 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::nio::ByteBuffer**

This class defines six categories of operations upon byte buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.639 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

Data Structures

- class **decaf::nio::CharBuffer**

This class defines four categories of operations upon character buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.640 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::DoubleBuffer**
This class defines four categories of operations upon double buffers:.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.641 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::FloatBuffer**

This class defines four categories of operations upon float buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.642 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::IntBuffer**

This class defines four categories of operations upon int buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.643 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::nio::InvalidMarkException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.644 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::LongBuffer**

This class defines four categories of operations upon long long buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.645 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::nio::ReadOnlyBufferException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.646 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::ShortBuffer**

This class defines four categories of operations upon short buffers:.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.647 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class **decaf::security::auth::x500::X500Principal**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::auth**
- namespace **decaf::security::auth::x500**

7.648 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector>
#include <decaf/util/Config.h>
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::Certificate**
Base interface for all identity certificates.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.649 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateEncodingException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.650 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::cert::CertificateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.651 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateExpiredException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.652 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateNotYetValidException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.653 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateParsingException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.654 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
```

Data Structures

- class **decaf::security::cert::X509Certificate**
Base interface for all identity certificates.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.655 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security::GeneralSecurityException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.656 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::InvalidKeyException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.657 src/main/decaf/security/Key.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Key**
*The **Key** (p. 1953) interface is the top-level interface for all keys.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.658 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::KeyException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.659 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchAlgorithmException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.660 src/main/decaf/security/NoSuchProviderException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchProviderException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.661 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Principal**

Base interface for a principal, which can represent an individual or organization.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.662 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::PublicKey**
A public key.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.663 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::SignatureException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.664 src/main/decaf/security_provider/SecurityProvider.h File Reference

Data Structures

- class `decaf::security_provider::SecurityProvider`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::security_provider`

7.665 src/main/decaf/security_provider/SecurityProviderMap.h File Reference

```
#include <map>
#include <vector>
#include <string>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security_provider::SecurityProviderMap**
Lookup Map for Connector Factories.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security_provider**

7.666 src/main/decaf/security_provider/SecurityProviderRegistrar.h File Reference

```
#include <string>
```

```
#include <decaf/security_provider/SecurityProviderMap.h>
```

Data Structures

- class **decaf::security_provider::SecurityProviderRegistrar**

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**

7.667 src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h File Reference

```
#include <decaf/security/auth/x500/X500Principal.h>
#include <openssl/x509.h>
```

Data Structures

- class **decaf::security_provider::unix::openssl::OpenSSLX500Principal**
The `OpenSSLX500Principal` (p. 2439) wraps around an `OpenSSL X509_NAME` structure.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security_provider**
- namespace **decaf::security_provider::unix**
- namespace **decaf::security_provider::unix::openssl**

7.668 src/main/decaf/security_provider/unix/openssl/OpenSSLX509C File Reference

```
#include <decaf/security/cert/X509Certificate.h>
```

Data Structures

- class **decaf::security_provider::unix::openssl::OpenSSLX509Certificate**

Namespaces

- namespace **decaf**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **decaf::security_provider**
- namespace **decaf::security_provider::unix**
- namespace **decaf::security_provider::unix::openssl**

7.669 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractCollection**< E >

*This class provides a skeletal implementation of the **Collection** (p. 1054) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.670 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractList**< **E** >

*This class provides a skeletal implementation of the **List** (p. 1984) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.671 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractMap**< **K**, **V**, **COMPARATOR** >

*This class provides a skeletal implementation of the **Map** (p. 2094) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.672 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractQueue**< **E** >
*This class provides skeletal implementations of some **Queue** (p. 2671) operations.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.673 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSequentialList**< **E** >

*This class provides a skeletal implementation of the **List** (p. 1984) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.674 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSet**< E >

*This class provides a skeletal implementation of the **Set** (p. 2905) interface to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.675 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

Data Structures

- class **decaf::util::Collection< E >**
The root interface in the collection hierarchy.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.676 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
#include <algorithm>
#include <functional>
```

Data Structures

- class **decaf::util::Comparator**< **T** >
A comparison function, which imposes a total ordering on some collection of objects.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.677 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

Data Structures

- class **decaf::util::comparators::Less**< **E** >
*Simple **Less** (p. 1981) **Comparator** (p. 1086) that compares to elements to determine if the first is less than the second.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::comparators**

7.678 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**

A boolean value that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.679 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**
An int value that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.680 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Long.h>
#include <apr_atomic.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference< T >**
An Pointer reference that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.681 src/main/decaf/util/concurrent/BlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.682 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.683 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::Callable**< **V** >
A task that returns a result and may throw an exception.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.684 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CancellationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.685 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

Defines

- **#define WAIT_INFINITE 0xFFFFFFFF**
The synchronized macro defines a mechanism for synchronizing a section of code.
- **#define synchronized(W)**

7.685.1 Define Documentation

7.685.1.1 #define synchronized(W)

Value:

```
if(false){}
else
    for( decaf::util::concurrent::Lock lock_W(W);
        lock_W.isLocked(); lock_W.unlock() )
```

7.685.1.2 #define WAIT_INFINITE 0xFFFFFFFF

The synchronized macro defines a mechanism for synchronizing a section of code. The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 4126) { // Do something that needs synchronizing. } }
```

7.686 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentMap**< **K**, **V**, **COMPARATOR** >
*Interface for a **Map** (p. 2094) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2094) interface.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.687 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR >
Map (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.688 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CountDownLatch**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.689 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

Data Structures

- class **decaf::util::concurrent::Delayed**
A mix-in style interface for marking objects that should be acted upon after a given delay.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.690 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutionException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.691 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::Executor**
*An object that executes submitted **decaf.lang.Runnable** (p. 2816) tasks.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.692 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutorService**

*An **Executor** (p. 1608) that provides methods to manage termination and methods that can produce a **Future** (p. 1701) for tracking progress of one or more asynchronous tasks.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.693 src/main/decaf/util/concurrent/Future.h File Reference

Data Structures

- class **decaf::util::concurrent::Future**< **V** >
*A **Future** (p. 1701) represents the result of an asynchronous computation.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.694 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.695 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Lock**

***Lock** (p. 2017) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.696 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Condition**

***Condition** (p. 1118) factors out the **Mutex** (p. 2385) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2017) implementations.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.697 src/main/decaf/util/concurrent/locks/LockSupport.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::locks::LockSupport**

Basic thread blocking primitives for creating locks and other synchronization classes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.698 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**

*A **ReadWriteLock** (p. 2688) maintains a pair of associated locks, one for read-only operations and one for writing.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.699 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**
*A reentrant mutual exclusion **Lock** (p. 2017) with extended capabilities.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.700 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Mutex**

***Mutex** (p. 2985) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.701 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThread**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.702 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThreadListener**
Abstract Listener Interface for users of `ThreadPool` (p. 3175).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.703 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.704 src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p.??).*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.705 src/main/decaf/util/concurrent/Semaphore.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <memory>
```

Data Structures

- class **decaf::util::concurrent::Semaphore**

A counting semaphore.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.706 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Synchronizable**

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.707 src/main/decaf/util/concurrent/SynchronousQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::SynchronousQueue< E >**
A BlockingQueue blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.
- class **decaf::util::concurrent::SynchronousQueue< E >::EmptyIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.708 src/main/decaf/util/concurrent/TaskListener.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::TaskListener**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.709 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ThreadFactory**
*public interface **ThreadFactory** (p. 3172)*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.710 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::ThreadPool**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.711 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::TimeoutException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.712 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::util::concurrent::TimeUnit**

*A **TimeUnit** (p. 3205) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.713 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::util::Date**
Wrapper class around a time value in milliseconds.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.714 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::util::Iterator**< **T** >
Defines an object that can be used to iterate over the elements of a collection.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.715 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

Data Structures

- class **decaf::util::List**< **E** >
An ordered collection (also known as a sequence).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.716 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::ListIterator**< **E** >

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.717 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/logging/StreamHandler.h>  
#include <decaf/io/StandardErrorOutputStream.h>
```

7.718 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::Filter**

*A **Filter** (p. 1630) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.719 src/main/decaf/util/logging/Formatter.h File Reference

Data Structures

- class **decaf::util::logging::Formatter**
*A **Formatter** (p. 1699) provides support for formatting *LogRecords*.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.720 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>
```

```
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::Handler**

*A **Handler** (p. 1709) object takes log messages from a **Logger** (p. 2028) and exports them.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.721 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <list>
#include <string>
#include <stdarg.h>
```

Data Structures

- class `decaf::util::logging::Logger`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::util`
- namespace `decaf::util::logging`

7.722 src/main/decaf/util/logging/LoggerCommon.h File Reference

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

Enumerations

- enum **decaf::util::logging::Level** {
 decaf::util::logging::Off, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**,
 decaf::util::logging::Debug,
 decaf::util::logging::Info, **decaf::util::logging::Warn**, **decaf::util::logging::Error**,
 decaf::util::logging::Fatal,
 decaf::util::logging::Throwing }
Defines an enumeration for logging levels.

7.723 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

Defines

- **#define LOGDECAF_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;**
- **#define LOGDECAF_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);**
- **#define LOGDECAF_DECLARE_LOCAL(loggerName) decaf::util::logging::Logger loggerName;**
- **#define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);**
- **#define LOGDECAF_DEBUG_1(logger, message, value)**
- **#define LOGDECAF_INFO(logger, message) logger.info(__FILE__, __LINE__, message);**
- **#define LOGDECAF_ERROR(logger, message) logger.error(__FILE__, __LINE__, message);**
- **#define LOGDECAF_WARN(logger, message) logger.warn(__FILE__, __LINE__, message);**
- **#define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE__, __LINE__, message);**

7.723.1 Define Documentation

7.723.1.1 #define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);

7.723.1.2 #define LOGDECAF_DEBUG_1(logger, message, value)

Value:

```
;
{
    std::ostringstream ostream;
    ostream << message << value;
    logger.debug(__FILE__, __LINE__, ostream.str());
}
```

- 7.723.1.3 `#define LOGDECAF_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;`
- 7.723.1.4 `#define LOGDECAF_DECLARE_LOCAL(loggerName) decaf::util::logging::Logger loggerName;`
- 7.723.1.5 `#define LOGDECAF_ERROR(logger, message) logger.error(__FILE__, __LINE__, message);`
- 7.723.1.6 `#define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE__, __LINE__, message);`
- 7.723.1.7 `#define LOGDECAF_INFO(logger, message) logger.info(__FILE__, __LINE__, message);`
- 7.723.1.8 `#define LOGDECAF_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);`
- 7.723.1.9 `#define LOGDECAF_WARN(logger, message) logger.warn(__FILE__, __LINE__, message);`

7.724 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

Data Structures

- class `decaf::util::logging::LoggerHierarchy`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::logging`

7.725 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p. 2045) object that is used to maintain a set of shared state about Loggers and log services.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.726 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::LogRecord**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.727 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::logging::LogWriter**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.728 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

Data Structures

- class **decaf::util::logging::MarkBlockLogger**
Defines a class that can be used to mark the entry and exit from scoped blocks.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.729 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 2657).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.730 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/logging/formatter.h>
```

Data Structures

- class **decaf::util::logging::SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 2050) in a human readable format.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.731 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::SimpleLogger**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.732 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/Filter.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::StreamHandler**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.733 src/main/decaf/util/Map.h File Reference

```
#include <functional>
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::Map**< **K**, **V**, **COMPARATOR** >
***Map** (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **decaf::util::Map**< **K**, **V**, **COMPARATOR** >::**Entry**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.734 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Collection.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/comparators/Less.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <memory>
```

Data Structures

- class **decaf::util::PriorityQueue< E >**
An unbounded priority queue based on a binary heap algorithm.
- class **decaf::util::PriorityQueue< E >::PriorityQueueIterator**
- class **decaf::util::PriorityQueue< E >::ConstPriorityQueueIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.735 src/main/decaf/util/Properties.h File Reference

```
#include <memory>
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Reader.h>
#include <decaf/io/Writer.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::Properties**
Java-like properties class for mapping string names to string values.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.736 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

Data Structures

- class **decaf::util::Random**

***Random** (p. 2677) Value Generator which is used to generate a stream of pseudorandom numbers.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.737 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

Data Structures

- class **decaf::util::Set**< **E** >
A collection that contains no duplicate elements.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.738 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

Data Structures

- class **decaf::util::StlList< E >**
List (p. 1984) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.
- class **decaf::util::StlList< E >::StlListIterator**
- class **decaf::util::StlList< E >::ConstStlListIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.739 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::StlMap< K, V, COMPARATOR >**
Map (p. 2094) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.740 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::StlQueue< T >**
*The **Queue** (p. 2671) class accepts messages with an **psuh(m)** command where *m* is the message to be queued.*
- class **decaf::util::StlQueue< T >::QueueIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.741 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

Data Structures

- class **decaf::util::StlSet< E >**
Set (p.2905) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.
- class **decaf::util::StlSet< E >::SetIterator**
- class **decaf::util::StlSet< E >::ConstSetIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.742 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::StringTokenizer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.743 src/main/decaf/util/Timer.h File Reference

```
#include <memory>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::Timer**

A facility for threads to schedule tasks for future execution in a background thread.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.744 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3188).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::util**

7.745 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

Data Structures

- class **decaf::util::UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3356)).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

Index

- ~AbstractCollection
 - decaf::util::AbstractCollection, 150
- ~AbstractList
 - decaf::util::AbstractList, 160
- ~AbstractMap
 - decaf::util::AbstractMap, 161
- ~AbstractQueue
 - decaf::util::AbstractQueue, 163
- ~AbstractSequentialList
 - decaf::util::AbstractSequentialList, 166
- ~AbstractSet
 - decaf::util::AbstractSet, 167
- ~AbstractTransportFactory
 - activemq::transport::AbstractTransportFactory, 169
- ~ActiveMQAckHandler
 - activemq::core::ActiveMQAckHandler, 171
- ~ActiveMQBlobMessage
 - activemq::commands::ActiveMQBlobMessage, 173
- ~ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 182
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 194
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 178
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 186
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 190
- ~ActiveMQBytesMessage
 - activemq::commands::ActiveMQBytesMessage, 200
- ~ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 218
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 230
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 214
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 222
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 226
- ~ActiveMQCPP
 - activemq::library::ActiveMQCPP, 272
- ~ActiveMQConnection
 - activemq::core::ActiveMQConnection, 235
- ~ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnectionFactory, 245
- ~ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnectionMetaData, 250
- ~ActiveMQConnectionSupport
 - activemq::core::ActiveMQConnectionSupport, 254
- ~ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 265
- ~ActiveMQDestination
 - activemq::commands::ActiveMQDestination, 276
- ~ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 290
 - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 302
 - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 286
 - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 294
 - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 298
- ~ActiveMQException
 - activemq::exceptions::ActiveMQException, 306
- ~ActiveMQMapMessage
 - activemq::commands::ActiveMQMapMessage, 311
- ~ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 327
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 339
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 323
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 331
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 335

- 335
- ~ActiveMQMessage
 - activemq::commands::ActiveMQMessage, 342
- ~ActiveMQMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 350
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 362
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 346
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 354
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 358
- ~ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 368
- ~ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 384
- ~ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 391
 - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 403
 - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 387
 - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 395
 - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 399
- ~ActiveMQProducer
 - activemq::core::ActiveMQProducer, 407
- ~ActiveMQProperties
 - activemq::util::ActiveMQProperties, 415
- ~ActiveMQQueue
 - activemq::commands::ActiveMQQueue, 420
- ~ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 428
 - activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 440
 - activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 424
 - activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 432
 - activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 436
- ~ActiveMQSession
 - activemq::core::ActiveMQSession, 447
- ~ActiveMQSessionExecutor
 - activemq::core::ActiveMQSessionExecutor, 461
- ~ActiveMQStreamMessage
 - activemq::commands::ActiveMQStreamMessage, 467
- ~ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 484
 - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 496
 - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 489
 - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 488
 - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 492
- ~ActiveMQTempDestination
 - activemq::commands::ActiveMQTempDestination, 500
- ~ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 508
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 520
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 504
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 512
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 516
- ~ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 524
- ~ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 533
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 545
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 529
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 537
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 544
- ~ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 549
- ~ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 558
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 570
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 554

- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 651
- 562 ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 665
- 566 ~BaseCommandMarshaller
- ~ActiveMQTextMessage
- activemq::commands::ActiveMQTextMessage, 686
- 574 ~BaseCommandMarshaller
- ~ActiveMQTextMessageMarshaller
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 658
- 583 ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 672
- 595 ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 679
- 579 ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 698
- 587 ~BaseDataStructure
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 716
- 591 ~BaseDataStructure
- ~ActiveMQTopic
- activemq::commands::ActiveMQTopic, 599
- ~ActiveMQTopicMarshaller
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 726
- 607 ~BaseDataStructure
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 733
- 619 ~BaseDataStructure
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 740
- 603 ~BaseDataStructure
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 747
- 611 ~BaseDataStructure
- activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 749
- 615 ~BaseDataStructure
- ~ActiveMQTransactionContext
- activemq::core::ActiveMQTransactionContext, 743
- 623 ~BaseDataStructure
- ~Appendable
- decaf::lang::Appendable, 626
- ~AprPool
- decaf::internal::AprPool, 628
- ~AtomicBoolean
- decaf::util::concurrent::atomic::AtomicBoolean, 630
- 630 ~BaseDataStructure
- ~AtomicInteger
- decaf::util::concurrent::atomic::AtomicInteger, 634
- 634 ~BaseDataStructure
- ~AtomicReference
- decaf::util::concurrent::atomic::AtomicReference, 642
- 642 ~BaseDataStructure
- ~BackupTransport
- activemq::transport::failover::BackupTransport, 644
- 644 ~BaseDataStructure
- ~BackupTransportPool
- activemq::transport::failover::BackupTransportPool, 648
- 648 ~BaseDataStructure
- ~BaseCommand
- activemq::commands::BaseCommand, 651
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 665
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 686
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 658
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 672
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 679
- ~BaseDataStructure
- activemq::wireformat::openwire::marshal::BaseDataStructureMarshaller, 698
- ~BaseDataStructure
- activemq::wireformat::openwire::marshal::BaseDataStructureMarshaller, 716
- ~BaseDataStructure
- ~BindException
- decaf::net::BindException, 722
- ~BlockingByteArrayInputStream
- decaf::io::BlockingByteArrayInputStream, 726
- ~Boolean
- decaf::lang::Boolean, 733
- ~BooleanExpression
- activemq::commands::BooleanExpression, 740
- ~BooleanStream
- activemq::wireformat::openwire::utils::BooleanStream, 749
- ~BrokenBarrierException
- decaf::util::concurrent::BrokenBarrierException, 743
- ~BrokerError
- activemq::commands::BrokerError, 746
- ~BrokerException
- activemq::exceptions::BrokerException, 749
- ~BrokerId
- activemq::commands::BrokerId, 752
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 760
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 772
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 756
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 764
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 768
- ~BrokerInfo
- activemq::commands::BrokerInfo, 777

- ~BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 788
 - activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 800
 - activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 784
 - activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 792
 - activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 796
- ~Buffer
 - decaf::nio::Buffer, 805
- ~BufferFactory
 - decaf::internal::nio::BufferFactory, 826
- ~BufferOverflowException
 - decaf::nio::BufferOverflowException, 835
- ~BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 838
- ~BufferedInputStream
 - decaf::io::BufferedInputStream, 810
- ~BufferedOutputStream
 - decaf::io::BufferedOutputStream, 815
- ~BufferedSocket
 - decaf::net::BufferedSocket, 818
- ~Byte
 - decaf::lang::Byte, 842
- ~ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 855
- ~ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 875
- ~ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 894
- ~ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 902
- ~ByteArrayPerspective
 - decaf::internal::nio::ByteArrayPerspective, 911
- ~ByteBuffer
 - decaf::nio::ByteBuffer, 918
- ~BytesMessage
 - cms::BytesMessage, 941
- ~CMSException
 - cms::CMSException, 1032
- ~CMSExceptionSupport
 - activemq::util::CMSExceptionSupport, 1034
- ~CMSProperties
 - cms::CMSProperties, 1037
- ~CMSSecurityException
 - cms::CMSSecurityException, 1040
- ~CachedConsumer
 - activemq::cmsutil::CachedConsumer, 954
- ~CachedProducer
 - activemq::cmsutil::CachedProducer, 958
- ~Callable
 - decaf::concurrent::Callable, 964
- ~CancellationException
 - decaf::concurrent::CancellationException, 966
- ~Certificate
 - decaf::security::cert::Certificate, 969
- ~CertificateEncodingException
 - decaf::security::cert::CertificateEncodingException, 973
- ~CertificateException
 - decaf::security::cert::CertificateException, 975
- ~CertificateExpiredException
 - decaf::security::cert::CertificateExpiredException, 977
- ~CertificateNotYetValidException
 - decaf::security::cert::CertificateNotYetValidException, 979
- ~CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 981
- ~CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 992
- ~CharBuffer
 - decaf::nio::CharBuffer, 1001
- ~CharSequence
 - decaf::lang::CharSequence, 1014
- ~ClassCastException
 - decaf::lang::exceptions::ClassCastException, 1017
- ~CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 1022
- ~Closeable
 - cms::Closeable, 1021
 - decaf::io::Closeable, 1019
- ~CmsAccessor
 - activemq::cmsutil::CmsAccessor, 1025
- ~CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 1029
- ~CmsTemplate
 - activemq::cmsutil::CmsTemplate, 1044
- ~Collection
 - decaf::util::Collection, 1056
- ~Command
 - activemq::commands::Command, 1064
- ~CommandVisitor
 - activemq::state::CommandVisitor, 1071
- ~CommandVisitorAdapter

- activemq::state::CommandVisitorAdapter, 1080
- ~Comparable
 - decaf::lang::Comparable, 1083
- ~Comparator
 - decaf::util::Comparator, 1086
- ~CompositeData
 - activemq::util::CompositeData, 1089
- ~CompositeTask
 - activemq::threads::CompositeTask, 1090
- ~CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 1093
- ~CompositeTransport
 - activemq::transport::CompositeTransport, 1095
- ~ConcurrentMap
 - decaf::util::concurrent::ConcurrentMap, 1098
- ~ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 1106
- ~Condition
 - decaf::util::concurrent::locks::Condition, 1120
- ~ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 1124
- ~ConnectException
 - decaf::net::ConnectException, 1129
- ~Connection
 - cms::Connection, 1132
- ~ConnectionControl
 - activemq::commands::ConnectionControl, 1136
- ~ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1145
 - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1157
 - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1141
 - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1149
 - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1153
- ~ConnectionError
 - activemq::commands::ConnectionError, 1161
- ~ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1169
 - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1181
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1165
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1173
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1177
- ~ConnectionFactory
 - cms::ConnectionFactory, 1185
- ~ConnectionId
 - activemq::commands::ConnectionId, 1188
- ~ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1196
 - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1208
 - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1192
 - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1200
 - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1204
- ~ConnectionInfo
 - activemq::commands::ConnectionInfo, 1212
- ~ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1226
 - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1234
 - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1218
 - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1222
 - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1230
- ~ConnectionMetaData
 - activemq::commands::ConnectionMetaData, 1238
- ~ConnectionState
 - activemq::transport::ConnectionState, 1242
- ~ConnectionStateTracker
 - activemq::transport::ConnectionStateTracker, 1246
- ~ConsumerControl
 - activemq::commands::ConsumerControl, 1251
- ~ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1264
 - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1272
 - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1256
 - activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1260

- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1268
- ~ConsumerId
 - activemq::commands::ConsumerId, 1276
- ~ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1289
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1297
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1281
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1285
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1293
- ~ConsumerInfo
 - activemq::commands::ConsumerInfo, 1302
- ~ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1314
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1326
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1310
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1318
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1322
- ~ConsumerState
 - activemq::state::ConsumerState, 1329
- ~ControlCommand
 - activemq::commands::ControlCommand, 1331
- ~ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1343
 - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1351
 - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1335
 - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1339
 - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1347
- ~CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1354
- ~DataArrayResponse
 - activemq::commands::DataArrayResponse, 1357
- ~DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1368
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1376
- activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1360
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1364
- activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1372
- ~DataInputStream
 - decaf::io::DataInputStream, 1381
- ~DataOutputStream
 - decaf::io::DataOutputStream, 1389
- ~DataResponse
 - activemq::commands::DataResponse, 1396
- ~DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1403
 - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1415
 - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1399
 - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1411
 - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1407
- ~DataStreamMarshaller
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1419
- ~DataStructure
 - activemq::commands::DataStructure, 1461
- ~Date
 - decaf::util::Date, 1468
- ~DecafRuntime
 - decaf::internal::DecafRuntime, 1472
- ~DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner, 1473
- ~DefaultTransportListener
 - activemq::transport::DefaultTransportListener, 1479
- ~Delayed
 - decaf::util::concurrent::Delayed, 1477
- ~DeliveryMode
 - cms::DeliveryMode, 1479
- ~Destination
 - cms::Destination, 1481
- ~DestinationInfo
 - activemq::commands::DestinationInfo, 1485
- ~DestinationInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1502
 - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1490

- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1494
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1598
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1498
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1602
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1506
- decaf::util::concurrent::ExecutionException, 1606
- ~DestinationResolver
 - activemq::cmsutil::DestinationResolver, 1509
- ~DiscoveryEvent
 - activemq::commands::DiscoveryEvent, 1512
- ~DiscoveryEventMarshaller
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1532
 - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1516
 - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1520
 - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1524
 - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1528
- ~Dispatcher
 - activemq::core::Dispatcher, 1536
- ~Double
 - decaf::lang::Double, 1539
- ~DoubleArrayBuffer
 - decaf::internal::nio::DoubleArrayBuffer, 1550
- ~DoubleBuffer
 - decaf::nio::DoubleBuffer, 1558
- ~DynamicDestinationResolver
 - activemq::cmsutil::DynamicDestinationResolver, 1568
- ~EOFException
 - decaf::io::EOFException, 1572
- ~Entry
 - decaf::util::Map::Entry, 1570
- ~Exception
 - decaf::lang::Exception, 1576
- ~ExceptionListener
 - cms::ExceptionListener, 1581
- ~ExceptionResponse
 - activemq::commands::ExceptionResponse, 1583
- ~ExceptionResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1590
 - activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1586
 - activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1594
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1598
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1602
- ~ExecutionException, 1606
- ~Executor
 - decaf::util::concurrent::Executor, 1609
- ~ExecutorService
 - decaf::util::concurrent::ExecutorService, 1611
- ~FailoverTransport
 - activemq::transport::failover::FailoverTransport, 1614
- ~FailoverTransportFactory
 - activemq::transport::failover::FailoverTransportFactory, 1625
- ~FailoverTransportListener
 - activemq::transport::failover::FailoverTransportListener, 1628
- ~Filter
 - decaf::util::logging::Filter, 1630
- ~FilterInputStream
 - decaf::io::FilterInputStream, 1633
- ~FilterOutputStream
 - decaf::io::FilterOutputStream, 1642
- ~Float
 - decaf::lang::Float, 1649
- ~FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1660
- ~FloatBuffer
 - decaf::nio::FloatBuffer, 1667
- ~FlushCommand
 - activemq::commands::FlushCommand, 1677
- ~FlushCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1684
 - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1680
 - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1688
 - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1692
 - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1696
- ~Formatter
 - decaf::util::logging::Formatter, 1699
- ~Future
 - decaf::util::concurrent::Future, 1702
- ~FutureResponse
 - activemq::transport::failover::FutureResponse, 1704

- ~GeneralSecurityException
 - decaf::security::GeneralSecurityException, 1707
- ~Handler
 - decaf::util::logging::Handler, 1710
- ~HexStringParser
 - decaf::internal::util::HexStringParser, 1713
- ~HexTable
 - activemq::wireformat::openwire::utils::HexTable, 1715
- ~HttpRequestException
 - decaf::net::HttpRequestException, 1718
- ~IOException
 - decaf::io::IOException, 1821
- ~IOTransport
 - activemq::transport::IOTransport, 1825
- ~IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1721
- ~IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1724
- ~IllegalStateException
 - cms::IllegalStateException, 1729
 - decaf::lang::exceptions::IllegalStateException, 1727
- ~IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1731
- ~InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1734
- ~IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1738
- ~InputStream
 - decaf::io::InputStream, 1741
- ~IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer, 1746
- ~IntBuffer
 - decaf::nio::IntBuffer, 1753
- ~Integer
 - decaf::lang::Integer, 1765
- ~IntegerResponse
 - activemq::commands::IntegerResponse, 1778
- ~IntegerResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1785
 - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1781
 - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1789
- activemq::wireformat::openwire::marshal::v4::IntegerResponse, 1793
- activemq::wireformat::openwire::marshal::v5::IntegerResponse, 1797
- ~InternalCommandListener
 - activemq::transport::mock::InternalCommandListener, 1800
- ~InterruptedException
 - decaf::lang::exceptions::InterruptedException, 1803
- ~InterruptedIOException
 - decaf::io::InterruptedIOException, 1806
- ~InvalidClientIdException
 - cms::InvalidClientIdException, 1808
- ~InvalidDestinationException
 - cms::InvalidDestinationException, 1809
- ~InvalidKeyException
 - decaf::security::InvalidKeyException, 1811
- ~InvalidMarkException
 - decaf::nio::InvalidMarkException, 1814
- ~InvalidSelectorException
 - cms::InvalidSelectorException, 1816
- ~InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 1818
- ~Iterable
 - decaf::lang::Iterable, 1830
- ~Iterator
 - decaf::util::Iterator, 1832
- ~JournalQueueAck
 - activemq::commands::JournalQueueAck, 1835
- ~JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 1851
 - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 1839
 - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 1847
 - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 1843
 - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 1855
- ~JournalTopicAck
 - activemq::commands::JournalTopicAck, 1859
- ~JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 1868
 - activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 1864
 - activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 1872

- activemq::wireformat::openwire::marshal::v4::JournalTopicAckIdMarshaller, 1876
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckIdMarshaller, 1880
- ~JournalTrace
 - activemq::commands::JournalTrace, 1884
- ~JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 1899
 - ~Less
 - activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 1887
 - ~List
 - activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 1895
 - ~ListIterator
 - activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 1891
 - ~LocalTransactionId
 - activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 1903
 - ~LocalTransactionIdMarshaller
- ~JournalTransaction
 - activemq::commands::JournalTransaction, 1907
- ~JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1923
 - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1911
 - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1915
 - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1919
 - ~Lock
 - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1927
 - decaf::util::concurrent::locks::Lock, 2018
- ~KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 1931
- ~KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 1950
 - decaf::util::logging::LogManager, 2047
 - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 1934
 - decaf::util::logging::LogRecord, 2051
 - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 1942
 - decaf::util::logging::LogWriter, 2055
 - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 1946
 - decaf::util::logging::Logger, 2030
 - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 1938
 - decaf::util::logging::LoggerHierarchy, 2037
- ~Key
 - decaf::security::Key, 1954
- ~KeyException
 - decaf::security::KeyException, 1956
- ~LastPartialCommand
 - activemq::commands::LastPartialCommand, 1959
- ~LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 1966
 - ~Long
 - ~LongArrayBuffer

- decaf::internal::nio::LongArrayBuffer, 2073
- ~LongBuffer
 - decaf::nio::LongBuffer, 2081
- ~LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 2090
- ~MalformedURLException
 - decaf::net::MalformedURLException, 2092
- ~Map
 - decaf::util::Map, 2095
- ~MapMessage
 - cms::MapMessage, 2108
- ~MarkBlockLogger
 - decaf::util::logging::MarkBlockLogger, 2117
- ~MarshalAware
 - activemq::wireformat::MarshalAware, 2118
- ~MarshallerFactory
 - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2123
 - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2121
 - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2124
 - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2122
 - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2125
- ~Math
 - decaf::lang::Math, 2128
- ~MemoryUsage
 - activemq::util::MemoryUsage, 2142
- ~Message
 - activemq::commands::Message, 2149
 - cms::Message, 2167
- ~MessageAck
 - activemq::commands::MessageAck, 2189
- ~MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2211
 - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2195
 - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2203
 - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2207
 - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2199
- ~MessageConsumer
 - cms::MessageConsumer, 2215
- ~MessageCreator
 - activemq::cmsutil::MessageCreator, 2218
- ~MessageDispatch
 - activemq::commands::MessageDispatch, 2220
- ~MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 2225
- ~MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2244
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2232
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2240
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2236
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2248
- ~MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 2252
- ~MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2270
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2258
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2266
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2262
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2274
- ~MessageEOFException
 - cms::MessageEOFException, 2277
- ~MessageFormatException
 - cms::MessageFormatException, 2278
- ~MessageId
 - activemq::commands::MessageId, 2280
- ~MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2301
 - activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2285
 - activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2293
 - activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2289
 - activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2297
- ~MessageListener
 - cms::MessageListener, 2304
- ~MessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2326
 - activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2306

- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2316
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2311
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2321
- ~MessageNotReadableException
 - cms::MessageNotReadableException, 2330
- ~MessageNotWriteableException
 - cms::MessageNotWriteableException, 2331
- ~MessageProducer
 - cms::MessageProducer, 2333
- ~MessagePropertyInterceptor
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2340
- ~MessagePull
 - activemq::commands::MessagePull, 2347
- ~MessagePullMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2360
 - activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2352
 - activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2356
 - activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2368
 - activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2364
- ~MockTransport
 - activemq::transport::mock::MockTransport, 2373
- ~MockTransportFactory
 - activemq::transport::mock::MockTransportFactory, 2383
- ~Mutex
 - decaf::util::concurrent::Mutex, 2386
- ~MutexHandle
 - decaf::util::concurrent::MutexHandle, 2390
- ~NetworkBridgeFilter
 - activemq::commands::NetworkBridgeFilter, 2394
- ~NetworkBridgeFilterMarshaller
 - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2410
 - activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2398
 - activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2402
 - activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2414
 - activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2406
- ~NoRouteToHostException
 - decaf::net::NoRouteToHostException, 2418
- ~NoSuchAlgorithmException
 - decaf::security::NoSuchAlgorithmException, 2421
- ~NoSuchElementException
 - decaf::lang::exceptions::NoSuchElementException, 2424
- ~NoSuchProviderException
 - decaf::security::NoSuchProviderException, 2427
- ~NullPointerException
 - decaf::lang::exceptions::NullPointerException, 2430
- ~Number
 - decaf::lang::Number, 2432
- ~NumberFormatException
 - decaf::lang::exceptions::NumberFormatException, 2437
- ~ObjectMessage
 - activemq::wireformat::openwire::marshal::v1::ObjectMessage, 2438
- ~OpenSSLX500Principal
 - decaf::util::provider::unix::openssl::OpenSSLX500Principal, 2440
- ~OpenSSLX509Certificate
 - decaf::util::provider::unix::openssl::OpenSSLX509Certificate, 2443
- ~OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2451
- ~OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2460
- ~OpenWireFormatNegotiator
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2463
- ~OpenWireResponseBuilder
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2466
- ~OpenwireStringSupport
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2468
- ~OutputStream
 - decaf::io::OutputStream, 2470
- ~PartialCommand
 - activemq::commands::PartialCommand, 2472
- ~PartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2490
 - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2478
- ~PartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2482

- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2486
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2494
- ~Pointer
 - decaf::lang::Pointer, 2500
- ~PooledSession
 - activemq::cmsutil::PooledSession, 2507
- ~PooledThread
 - decaf::util::concurrent::PooledThread, 2518
- ~PooledThreadListener
 - decaf::util::concurrent::PooledThreadListener, 2520
- ~PortUnreachableException
 - decaf::net::PortUnreachableException, 2523
- ~PrimitiveList
 - activemq::util::PrimitiveList, 2527
- ~PrimitiveMap
 - activemq::util::PrimitiveMap, 2538
- ~PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2547
- ~PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2553
- ~PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2561
- ~Principal
 - decaf::security::Principal, 2569
- ~PriorityQueue
 - decaf::util::PriorityQueue, 2574
- ~ProducerAck
 - activemq::commands::ProducerAck, 2580
- ~ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2600
 - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2584
 - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2588
 - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2592
 - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2596
- ~ProducerCallback
 - activemq::cmsutil::ProducerCallback, 2603
- ~ProducerExecutor
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2604
- ~ProducerId
 - activemq::commands::ProducerId, 2607
- ~ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2624
 - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2612
 - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2616
 - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2628
 - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2620
- ~ProducerInfo
 - activemq::commands::ProducerInfo, 2632
- ~ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2649
 - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2637
 - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2653
 - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2645
 - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2641
- ~ProducerState
 - activemq::state::ProducerState, 2656
- ~Properties
 - decaf::util::Properties, 2659
- ~PropertiesChangeListener
 - decaf::util::logging::PropertiesChangeListener, 2666
- ~ProtocolException
 - decaf::net::ProtocolException, 2668
- ~PublicKey
 - decaf::security::PublicKey, 2670
- ~Queue
 - activemq::util::Queue, 2671
 - decaf::util::Queue, 2672
- ~QueueBrowserMarshaller
 - cms::QueueBrowser, 2675
- ~ReadCheckerMarshaller
 - activemq::transport::inactivity::ReadChecker, 2682
- ~ReadOnlyBufferException
 - decaf::io::ReadOnlyBufferException, 2686
- ~ReadWriteLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2689
- ~Reader
 - decaf::io::Reader, 2683
- ~ReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2690
- ~ReentrantLock

- decaf::util::concurrent::locks::ReentrantLock, 2693
- ~RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2700
- ~RejectedExecutionHandler
 - decaf::util::concurrent::RejectedExecutionHandler, 2702
- ~RemoveInfo
 - activemq::commands::RemoveInfo, 2704
- ~RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 2708
 - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2712
 - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2716
 - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 2720
 - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2724
- ~RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 2728
- ~RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 2741
 - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 2737
 - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 2745
 - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 2749
 - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2733
- ~ReplayCommand
 - activemq::commands::ReplayCommand, 2753
- ~ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 2773
 - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2757
 - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2761
 - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2769
 - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2765
- ~ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2776
- ~ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2777
- ~ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 2778
- ~Response
 - activemq::commands::Response, 2782
- ~ResponseBuilder
 - activemq::transport::mock::ResponseBuilder, 2785
- ~ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 2788
- ~ResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 2807
 - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 2792
 - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 2797
 - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 2812
 - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2802
- ~Runnable
 - decaf::lang::Runnable, 2816
- ~Runtime
 - decaf::lang::Runtime, 2817
- ~RuntimeException
 - decaf::lang::exceptions::RuntimeException, 2820
- ~SecurityProvider
 - decaf::security_provider::SecurityProvider, 2822
- ~SecurityProviderRegistrar
 - decaf::security_provider::SecurityProviderRegistrar, 2825
- ~Semaphore
 - decaf::util::concurrent::Semaphore, 2830
- ~SendCommand
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2833
- ~ServerSocket
 - decaf::net::ServerSocket, 2837
- ~Session
 - activemq::commands::Session, 2842
- ~SessionCallback
 - activemq::commands::SessionCallback, 2852
- ~SessionId
 - activemq::commands::SessionId, 2854
- ~SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 2862

- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2965
- 2866
- ~SocketException
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshallerException, 2973
- 2870
- ~SocketFactory
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshallerFactory, 2975
- 2874
- ~SocketInputStream
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshallerInputStream, 2978
- 2858
- ~SocketOutputStream
- decaf::net::SocketOutputStream, 2985
- ~SessionInfo
- activemq::commands::SessionInfo, 2878
- ~SocketTimeoutException
- ~SessionInfoMarshaller
- decaf::net::SocketTimeoutException, 2991
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 2882
- decaf::internal::io::StandardErrorOutputStream, 2995
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2890
- ~StandardInputStream
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2898
- decaf::io::StandardInputStream, 3001
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2886
- decaf::internal::io::StandardOutputStream, 3000
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2894
- ~Startable
- ~SessionPool
- cms::Startable, 3014
- activemq::cmsutil::SessionPool, 2901
- ~StaticInitializer
- ~SessionState
- activemq::core::ActiveMQConstants::StaticInitializer, 3016
- activemq::state::SessionState, 2904
- ~StlList
- ~Set
- decaf::util::Set, 2905
- decaf::util::StlList, 3022
- ~Short
- decaf::lang::Short, 2908
- ~StlMap
- decaf::util::StlMap, 3034
- ~ShortArrayBuffer
- decaf::internal::nio::ShortArrayBuffer, 2917
- ~StlQueue
- decaf::util::StlQueue, 3045
- ~ShortBuffer
- decaf::nio::ShortBuffer, 2925
- ~StlSet
- decaf::util::StlSet, 3053
- ~ShutdownInfo
- activemq::wireformat::stomp::StompFrame, 3062
- activemq::commands::ShutdownInfo, 2935
- ~StompHelper
- activemq::wireformat::stomp::StompHelper, 3067
- ~ShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 2938
- ~StompWireFormat
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 2950
- activemq::wireformat::stomp::StompWireFormat, 3072
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 2946
- ~StompWireFormatFactory
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 2942
- activemq::wireformat::stomp::StompWireFormatFactory, 3075
- ~Stoppable
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2954
- cms::Stoppable, 3076
- ~SignatureException
- decaf::security::SignatureException, 2958
- ~StreamHandler
- decaf::util::logging::StreamHandler, 3078
- ~SimpleFormatter
- decaf::util::logging::SimpleFormatter, 2960
- ~StreamMessage
- cms::StreamMessage, 3084
- ~SimpleLogger
- decaf::util::logging::SimpleLogger, 2962
- ~StringTokenizer
- decaf::util::StringTokenizer, 3095
- ~Socket
- ~SubscriptionInfo

- activemq::commands::SubscriptionInfo, 3098
- ~SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3103
 - activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3119
 - activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3111
 - activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3107
 - activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3115
- ~Synchronizable
 - decaf::util::concurrent::Synchronizable, 3123
- ~SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 3134
- ~Synchronization
 - activemq::core::Synchronization, 3137
- ~SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 3140
- ~System
 - decaf::lang::System, 3145
- ~Task
 - activemq::threads::Task, 3148
- ~TaskListener
 - decaf::util::concurrent::TaskListener, 3149
- ~TaskRunner
 - activemq::threads::TaskRunner, 3150
- ~TcpSocket
 - decaf::net::TcpSocket, 3154
- ~TcpTransport
 - activemq::transport::tcp::TcpTransport, 3161
- ~TcpTransportFactory
 - activemq::transport::tcp::TcpTransportFactory, 3164
- ~TemporaryQueue
 - cms::TemporaryQueue, 3166
- ~TemporaryTopic
 - cms::TemporaryTopic, 3168
- ~TextMessage
 - cms::TextMessage, 3170
- ~ThreadFactory
 - decaf::util::concurrent::ThreadFactory, 3172
- ~ThreadGroup
 - decaf::lang::ThreadGroup, 3174
- ~ThreadPool
 - decaf::util::concurrent::ThreadPool, 3177
- ~Throwable
 - decaf::lang::Throwable, 3182
- ~TimeUnit
 - decaf::util::concurrent::TimeUnit, 3207
- ~TimeoutException
 - decaf::util::concurrent::TimeoutException, 3186
- ~Timer
 - decaf::util::Timer, 3190
- ~TimerTask
 - decaf::util::TimerTask, 3200
- ~TimerTaskHeap
 - decaf::util::TimerTaskHeap, 3203
- ~Topic
 - cms::Topic, 3215
- ~Tracked
 - activemq::state::Tracked, 3216
- ~TransactionId
 - activemq::commands::TransactionId, 3218
- ~TransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3221
 - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3225
 - activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3237
 - activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3229
 - activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3233
- ~TransactionInfo
 - activemq::commands::TransactionInfo, 3241
- ~TransactionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3246
 - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3262
 - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3254
 - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3250
 - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3258
- ~TransactionState
 - activemq::state::TransactionState, 3266
- ~TransferQueue
 - decaf::internal::util::concurrent::TransferQueue, 3268
- ~TransferStack
 - decaf::internal::util::concurrent::TransferStack, 3271
- ~Transport
 - activemq::transport::Transport, 3274
- ~TransportFactory

- activemq::transport::TransportFactory, 3279
- ~TransportFilter
 - activemq::transport::TransportFilter, 3283
- ~TransportListener
 - activemq::transport::TransportListener, 3289
- ~TransportRegistry
 - activemq::transport::TransportRegistry, 3292
- ~URI
 - decaf::net::URI, 3311
- ~URIEncoderDecoder
 - decaf::internal::net::URIEncoderDecoder, 3319
- ~URIHelper
 - decaf::internal::net::URIHelper, 3323
- ~URIPool
 - activemq::transport::failover::URIPool, 3329
- ~URISyntaxException
 - decaf::net::URISyntaxException, 3337
- ~URIType
 - decaf::internal::net::URIType, 3341
- ~URL
 - decaf::net::URL, 3348
- ~URLDecoder
 - decaf::net::URLDecoder, 3349
- ~URLEncoder
 - decaf::net::URLEncoder, 3350
- ~UTFDataFormatException
 - decaf::io::UTFDataFormatException, 3355
- ~UUID
 - decaf::util::UUID, 3358
- ~UnknownHostException
 - decaf::net::UnknownHostException, 3295
- ~UnknownServiceException
 - decaf::net::UnknownServiceException, 3298
- ~UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 3302
- ~UnsupportedOperationException
 - cms::UnsupportedOperationException, 3306
 - decaf::lang::exceptions::UnsupportedOperationException, 3304
- ~Usage
 - activemq::util::Usage, 3351
- ~WireFormat
 - activemq::wireformat::WireFormat, 3364
- ~WireFormatFactory
 - activemq::wireformat::WireFormatFactory, 3367
- ~WireFormatInfo
 - activemq::commands::WireFormatInfo, 3371
- ~WireFormatInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3396
 - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3388
 - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3380
 - activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3392
 - activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3384
- ~WireFormatNegotiator
 - activemq::wireformat::WireFormatNegotiator, 3399
- ~WireFormatRegistry
 - activemq::wireformat::WireFormatRegistry, 3401
- ~WriteChecker
 - activemq::transport::inactivity::WriteChecker, 3403
- ~Writer
 - decaf::io::Writer, 3404
- ~X500Principal
 - decaf::security::auth::x500::X500Principal, 3406
- ~X509Certificate
 - decaf::security::cert::X509Certificate, 3407
- ~XATransactionId
 - activemq::commands::XATransactionId, 3411
- ~XATransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3420
 - activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3428
 - activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3416
 - activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3424
 - activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3432
- _FALSE
 - decaf::lang::Boolean, 736
- _TRUE
 - decaf::lang::Boolean, 736
- _array
 - decaf::internal::nio::CharArrayBuffer, 997
- _capacity
 - decaf::nio::Buffer, 808
- _limit
 - decaf::nio::Buffer, 808

- `_mark`
 - `decaf::nio::Buffer`, 808
- `_markSet`
 - `decaf::nio::Buffer`, 808
- `_position`
 - `decaf::nio::Buffer`, 808
- ABORT
 - `activemq::wireformat::stomp::StompCommandConstants`, 3059
- abs
 - `decaf::lang::Math`, 2128, 2129
- AbstractQueue
 - `decaf::util::AbstractQueue`, 163
- accept
 - `decaf::net::ServerSocket`, 2838
- ACK
 - `activemq::wireformat::stomp::StompCommandConstants`, 3059
- ACK_AUTO
 - `activemq::wireformat::stomp::StompCommandConstants`, 3059
- ACK_CLIENT
 - `activemq::wireformat::stomp::StompCommandConstants`, 3059
- ACK_INDIVIDUAL
 - `activemq::wireformat::stomp::StompCommandConstants`, 3059
- ACK_TYPE_CONSUMED
 - `activemq::core::ActiveMQConstants`, 261
- ACK_TYPE_DELIVERED
 - `activemq::core::ActiveMQConstants`, 261
- ACK_TYPE_INDIVIDUAL
 - `activemq::core::ActiveMQConstants`, 261
- ACK_TYPE_POISON
 - `activemq::core::ActiveMQConstants`, 261
- ACK_TYPE_REDELIVERED
 - `activemq::core::ActiveMQConstants`, 261
- acknowledge
 - `activemq::commands::ActiveMQMessageTemplates`, 368
 - `activemq::core::ActiveMQConsumer`, 265
 - `activemq::core::ActiveMQSession`, 447
 - `cms::Message`, 2167
- acknowledgeMessage
 - `activemq::core::ActiveMQAckHandler`, 171
- AcknowledgeMode
 - `cms::Session`, 2842
- AckType
 - `activemq::core::ActiveMQConstants`, 261
- ackType
 - `activemq::commands::MessageAck`, 2193
- acquire
 - `decaf::util::concurrent::Semaphore`, 2830
- acquireUninterruptibly
 - `decaf::util::concurrent::Semaphore`, 2831
- action
 - `activemq::cmsutil::CmsTemplate::ProducerExecutor`, 2605
- activemq, 69
- `activemq/exceptions/ExceptionDefines.h`
 - `AMQ_CATCH_EXCEPTION_-CONVERT`, 3533
 - `AMQ_CATCH_NOTHROW`, 3533
 - `AMQ_CATCH_RETHROW`, 3534
 - `AMQ_CATCHALL_NOTHROW`, 3534
 - `AMQ_CATCHALL_THROW`, 3534
- `activemq/util/Config.h`
 - `AMQCPP_API`, 3589
 - `HAVE_PTHREAD_H`, 3589
 - `HAVE_UUID_T`, 3589
 - `HAVE_UUID_UUID_H`, 3589
- `activemq::cmsutil`, 70
- `activemq::cmsutil::CachedConsumer`, 953
 - `asCachedConsumer`, 954
 - `CachedConsumer`, 954
 - `close`, 954
 - `setMessageListener`, 954
 - `getMessageSelector`, 954
 - `receive`, 954, 955
 - `retainNoWait`, 955
 - `setMessageListener`, 955
- `activemq::cmsutil::CachedProducer`, 957
 - `~CachedProducer`, 958
 - `CachedProducer`, 958
 - `close`, 958
 - `getDeliveryMode`, 958
 - `getDisableMessageID`, 958
 - `getDisableMessageTimeStamp`, 959
 - `getPriority`, 959
 - `getTimeToLive`, 959
 - `send`, 960, 961
 - `setDeliveryMode`, 961
 - `setDisableMessageID`, 962
 - `setDisableMessageTimeStamp`, 962
 - `setPriority`, 962
 - `setTimeToLive`, 963
- `activemq::cmsutil::CmsAccessor`, 1024
 - `~CmsAccessor`, 1025
 - `checkConnectionFactory`, 1025
 - `CmsAccessor`, 1025
 - `createConnection`, 1025
 - `createSession`, 1025
 - `destroy`, 1026
 - `getConnectionFactory`, 1026
 - `getResourceLifecycleManager`, 1026
 - `getSessionAcknowledgeMode`, 1026
 - `init`, 1027

- setConnectionFactory, 1027
- setSessionAcknowledgeMode, 1027
- activemq::cmsutil::CmsDestinationAccessor, 1028
 - ~CmsDestinationAccessor, 1029
 - checkDestinationResolver, 1029
 - CmsDestinationAccessor, 1029
 - destroy, 1029
 - getDestinationResolver, 1029
 - init, 1029
 - isPubSubDomain, 1029
 - resolveDestinationName, 1030
 - setDestinationResolver, 1030
 - setPubSubDomain, 1030
- activemq::cmsutil::CmsTemplate, 1041
 - ~CmsTemplate, 1044
 - CmsTemplate, 1044
 - DEFAULT_PRIORITY, 1053
 - DEFAULT_TIME_TO_LIVE, 1053
 - destroy, 1044
 - execute, 1044, 1045
 - getDefaultDestination, 1045, 1046
 - getDefaultDestinationName, 1046
 - getDeliveryMode, 1046
 - getPriority, 1046
 - getReceiveTimeout, 1046
 - getTimeToLive, 1046
 - init, 1046
 - isExplicitQosEnabled, 1047
 - isMessageIdEnabled, 1047
 - isMessageTimestampEnabled, 1047
 - isNoLocal, 1047
 - ProducerExecutor, 1053
 - receive, 1047, 1048
 - RECEIVE_TIMEOUT_INDEFINITE_WAIT, 1053
 - RECEIVE_TIMEOUT_NO_WAIT, 1053
 - ReceiveExecutor, 1053
 - receiveSelected, 1048, 1049
 - ResolveProducerExecutor, 1053
 - ResolveReceiveExecutor, 1053
 - send, 1049, 1050
 - SendExecutor, 1053
 - setDefaultDestination, 1050
 - setDefaultDestinationName, 1050
 - setDeliveryMode, 1050
 - setDeliveryPersistent, 1051
 - setExplicitQosEnabled, 1051
 - setMessageIdEnabled, 1051
 - setMessageTimestampEnabled, 1052
 - setNoLocal, 1052
 - setPriority, 1052
 - setPubSubDomain, 1052
 - setReceiveTimeout, 1052
 - setTimeToLive, 1052
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 2604
 - ~ProducerExecutor, 2604
 - action, 2605
 - destination, 2605
 - doInCms, 2604
 - getDestination, 2604
 - parent, 2605
 - ProducerExecutor, 2604
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2690
 - ~ReceiveExecutor, 2690
 - destination, 2691
 - doInCms, 2690
 - getDestination, 2691
 - getMessage, 2691
 - message, 2691
 - noLocal, 2691
 - parent, 2691
 - ReceiveExecutor, 2690
 - selector, 2691
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2776
 - ~ResolveProducerExecutor, 2776
 - getDestination, 2776
 - ResolveProducerExecutor, 2776
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2777
 - ~ResolveReceiveExecutor, 2777
 - getDestination, 2777
 - ResolveReceiveExecutor, 2777
- activemq::cmsutil::CmsTemplate::SendExecutor, 2836
 - ~SendExecutor, 2836
 - doInCms, 2836
 - SendExecutor, 2836
- activemq::cmsutil::DestinationResolver, 1509
 - ~DestinationResolver, 1509
 - destroy, 1509
 - init, 1509
 - resolveDestinationName, 1510
- activemq::cmsutil::DynamicDestinationResolver, 1568
 - ~DynamicDestinationResolver, 1568
 - destroy, 1568
 - init, 1568
 - resolveDestinationName, 1569
- activemq::cmsutil::MessageCreator, 2218
 - ~MessageCreator, 2218
 - createMessage, 2218
- activemq::cmsutil::PooledSession, 2505
 - ~PooledSession, 2507
 - close, 2507

- commit, 2507
- createBrowser, 2508
- createBytesMessage, 2508, 2509
- createCachedConsumer, 2509
- createCachedProducer, 2509
- createConsumer, 2510, 2511
- createDurableConsumer, 2511
- createMapMessage, 2512
- createMessage, 2512
- createProducer, 2512
- createQueue, 2513
- createStreamMessage, 2513
- createTemporaryQueue, 2513
- createTemporaryTopic, 2514
- createTextMessage, 2514
- createTopic, 2515
- getAcknowledgeMode, 2515
- getSession, 2515
- isTransacted, 2516
- PooledSession, 2507
- recover, 2516
- rollback, 2516
- unsubscribe, 2517
- activemq::cmsutil::ProducerCallback, 2603
 - ~ProducerCallback, 2603
 - doInCms, 2603
- activemq::cmsutil::ResourceLifecycleManager, 2778
 - ~ResourceLifecycleManager, 2778
 - addConnection, 2779
 - addDestination, 2779
 - addMessageConsumer, 2779
 - addMessageProducer, 2779
 - addSession, 2779
 - destroy, 2779
 - releaseAll, 2780
 - ResourceLifecycleManager, 2778
- activemq::cmsutil::SessionCallback, 2852
 - ~SessionCallback, 2852
 - doInCms, 2852
- activemq::cmsutil::SessionPool, 2901
 - ~SessionPool, 2901
 - getResourceLifecycleManager, 2902
 - returnSession, 2902
 - SessionPool, 2901
 - takeSession, 2902
- activemq::commands, 71
- activemq::commands::ActiveMQBlobMessage, 172
 - ~ActiveMQBlobMessage, 173
 - ActiveMQBlobMessage, 173
 - BINARY_MIME_TYPE, 176
 - clone, 173
 - cloneDataStructure, 173
 - copyDataStructure, 173
 - equals, 174
 - getDataStructureType, 174
 - getMimeType, 174
 - getName, 174
 - getRemoteBlobUrl, 174
 - ID_ACTIVEMQBLOBMESSAGE, 176
 - isDeletedByBroker, 175
 - setDeletedByBroker, 175
 - setMimeType, 175
 - setName, 175
 - setRemoteBlobUrl, 175
 - toString, 175
- activemq::commands::ActiveMQBytesMessage, 197
 - ~ActiveMQBytesMessage, 200
 - ActiveMQBytesMessage, 200
 - clearBody, 200
 - clone, 200
 - cloneDataStructure, 200
 - copyDataStructure, 201
 - equals, 201
 - getBodyBytes, 201
 - getBodyLength, 201
 - getDataStructureType, 202
 - ID_ACTIVEMQBYTESMESSAGE, 212
 - onSend, 202
 - readBoolean, 202
 - readByte, 202
 - readBytes, 203
 - readChar, 204
 - readDouble, 204
 - readFloat, 205
 - readInt, 205
 - readLong, 205
 - readShort, 206
 - readString, 206
 - readUnsignedShort, 206
 - readUTF, 207
 - reset, 207
 - setBodyBytes, 207
 - toString, 208
 - writeBoolean, 208
 - writeByte, 208
 - writeBytes, 209
 - writeChar, 209
 - writeDouble, 210
 - writeFloat, 210
 - writeInt, 210
 - writeLong, 211
 - writeShort, 211
 - writeString, 211
 - writeUnsignedShort, 211
 - writeUTF, 212

- activemq::commands::ActiveMQDestination, 274
 - ~ActiveMQDestination, 276
 - ActiveMQDestination, 276
 - advisory, 283
 - ADVISORY_PREFIX, 283
 - cloneDataStructure, 276
 - COMPOSITE_SEPARATOR, 283
 - CONNECTION_ADVISORY_PREFIX, 283
 - CONSUMER_ADVISORY_PREFIX, 283
 - copyDataStructure, 277
 - createDestination, 277
 - createTemporaryName, 277
 - DEFAULT_ORDERED_TARGET, 283
 - equals, 277
 - exclusive, 283
 - getClientId, 278
 - getCMSDestination, 278
 - getDataStructureType, 278
 - getDestinationType, 279
 - getOptions, 279
 - getOrderedTarget, 279
 - getPhysicalName, 279
 - ID_ACTIVEMQDESTINATION, 284
 - isAdvisory, 280
 - isComposite, 280
 - isConnectionAdvisory, 280
 - isConsumerAdvisory, 280
 - isExclusive, 280
 - isOrdered, 280
 - isProducerAdvisory, 280
 - isQueue, 281
 - isTemporary, 281
 - isTopic, 281
 - isWildcard, 281
 - options, 284
 - ordered, 284
 - orderedTarget, 284
 - physicalName, 284
 - PRODUCER_ADVISORY_PREFIX, 284
 - QUEUE_QUALIFIED_PREFIX, 284
 - setAdvisory, 281
 - setExclusive, 281
 - setOrdered, 282
 - setOrderedTarget, 282
 - setPhysicalName, 282
 - TEMP_POSTFIX, 284
 - TEMP_PREFIX, 284
 - TEMP_QUEUE_QUALIFIED_PREFIX, 284
 - TEMP_TOPIC_QUALIFIED_PREFIX, 284
 - TOPIC_QUALIFIED_PREFIX, 284
 - toString, 282
- activemq::commands::ActiveMQDestination::DestinationFilter, 1483
 - ANY_CHILD, 1483
 - ANY_DESCENDENT, 1483
- activemq::commands::ActiveMQMapMessage, 308
 - ~ActiveMQMapMessage, 311
 - ActiveMQMapMessage, 311
 - beforeMarshal, 311
 - checkMapIsUnmarshalled, 311
 - clearBody, 311
 - clone, 311
 - cloneDataStructure, 311
 - copyDataStructure, 312
 - equals, 312
 - getBoolean, 312
 - getByte, 312
 - getBytes, 313
 - getChar, 313
 - getDataStructureType, 313
 - getDouble, 314
 - getFloat, 314
 - getInt, 314
 - getLong, 315
 - getMap, 315
 - getMapNames, 315
 - getShort, 315
 - getString, 316
 - ID_ACTIVEMQMAPMESSAGE, 320
 - isMarshalAware, 316
 - itemExists, 316
 - setBoolean, 317
 - setByte, 317
 - setBytes, 317
 - setChar, 318
 - setDouble, 318
 - setFloat, 318
 - setInt, 319
 - setLong, 319
 - setShort, 319
 - setString, 320
 - toString, 320
- activemq::commands::ActiveMQMessage, 342
 - ~ActiveMQMessage, 342
 - ActiveMQMessage, 342
 - clone, 342
 - cloneDataStructure, 343
 - copyDataStructure, 343
 - equals, 343
 - getDataStructureType, 343
 - ID_ACTIVEMQMESSAGE, 344
 - toString, 343

- activemq::commands::ActiveMQMessageTemplate, 365
 - ~ActiveMQMessageTemplate, 368
 - acknowledge, 368
 - ActiveMQMessageTemplate, 368
 - clearBody, 368
 - clearProperties, 369
 - equals, 369
 - failIfReadOnlyBody, 369
 - failIfReadOnlyProperties, 369
 - failIfWriteOnlyBody, 369
 - getBooleanProperty, 369
 - getByteProperty, 370
 - getCMSCorrelationID, 370
 - getCMSDeliveryMode, 370
 - getCMSDestination, 371
 - getCMSExpiration, 371
 - getCMSMessageID, 371
 - getCMSPriority, 371
 - getCMSRedelivered, 372
 - getCMSReplyTo, 372
 - getCMSTimestamp, 372
 - getCMSType, 373
 - getDoubleProperty, 373
 - getFloatProperty, 373
 - getIntProperty, 374
 - getLongProperty, 374
 - getPropertyNames, 374
 - getShortProperty, 375
 - getStringProperty, 375
 - onSend, 375
 - propertyExists, 376
 - setBooleanProperty, 376
 - setByteProperty, 376
 - setCMSCorrelationID, 377
 - setCMSDeliveryMode, 377
 - setCMSDestination, 377
 - setCMSExpiration, 377
 - setCMSMessageID, 378
 - setCMSPriority, 378
 - setCMSRedelivered, 378
 - setCMSReplyTo, 379
 - setCMSTimestamp, 379
 - setCMSType, 379
 - setDoubleProperty, 379
 - setFloatProperty, 380
 - setIntProperty, 380
 - setLongProperty, 380
 - setShortProperty, 381
 - setStringProperty, 381
- activemq::commands::ActiveMQObjectMessage, 383
 - ~ActiveMQObjectMessage, 384
 - ActiveMQObjectMessage, 384
 - clone, 384
 - cloneDataStructure, 384
 - copyDataStructure, 384
 - equals, 384
 - getDataStructureType, 385
 - ID_ACTIVEMQOBJECTMESSAGE, 385
 - toString, 385
- activemq::commands::ActiveMQQueue, 419
 - ~ActiveMQQueue, 420
 - ActiveMQQueue, 420
 - clone, 420
 - cloneDataStructure, 420
 - copy, 420
 - copyDataStructure, 420
 - equals, 421
 - getCMSDestination, 421
 - getCMSProperties, 421
 - getDataStructureType, 421
 - getDestinationType, 421
 - getQueueName, 422
 - ID_ACTIVEMQQUEUE, 422
 - toString, 422
- activemq::commands::ActiveMQStreamMessage, 464
 - ~ActiveMQStreamMessage, 467
 - ActiveMQStreamMessage, 467
 - clearBody, 467
 - clone, 467
 - cloneDataStructure, 467
 - copyDataStructure, 467
 - equals, 468
 - getDataStructureType, 468
 - ID_ACTIVEMQSTREAMMESSAGE, 478
 - onSend, 468
 - readBoolean, 468
 - readByte, 469
 - readBytes, 469, 470
 - readChar, 470
 - readDouble, 471
 - readFloat, 471
 - readInt, 471
 - readLong, 472
 - readShort, 472
 - readString, 473
 - readUnsignedShort, 473
 - reset, 473
 - toString, 474
 - writeBoolean, 474
 - writeByte, 474
 - writeBytes, 475
 - writeChar, 475
 - writeDouble, 476
 - writeFloat, 476

- writeInt, 476
- writeLong, 476
- writeShort, 477
- writeString, 477
- writeUnsignedShort, 477
- activemq::commands::ActiveMQTempDestination, 499
 - ~ActiveMQTempDestination, 500
 - ActiveMQTempDestination, 500
 - cloneDataStructure, 500
 - close, 500
 - connection, 502
 - copyDataStructure, 500
 - equals, 500
 - getDataStructureType, 501
 - ID_ACTIVEMQTEMPDESTINATION, 502
 - setConnection, 501
 - toString, 501
- activemq::commands::ActiveMQTempQueue, 523
 - ~ActiveMQTempQueue, 524
 - ActiveMQTempQueue, 524
 - clone, 524
 - cloneDataStructure, 524
 - copy, 524
 - copyDataStructure, 524
 - destroy, 525
 - equals, 525
 - getCMSDestination, 525
 - getCMSProperties, 525
 - getDataStructureType, 526
 - getDestinationType, 526
 - getQueueName, 526
 - ID_ACTIVEMQTEMPQUEUE, 527
 - toString, 526
- activemq::commands::ActiveMQTempTopic, 548
 - ~ActiveMQTempTopic, 549
 - ActiveMQTempTopic, 549
 - clone, 549
 - cloneDataStructure, 549
 - copy, 549
 - copyDataStructure, 549
 - destroy, 550
 - equals, 550
 - getCMSDestination, 550
 - getCMSProperties, 550
 - getDataStructureType, 551
 - getDestinationType, 551
 - getTopicName, 551
 - ID_ACTIVEMQTEMPTOPIC, 552
 - toString, 551
- activemq::commands::ActiveMQTextMessage, 573
 - ~ActiveMQTextMessage, 574
 - ActiveMQTextMessage, 574
 - beforeMarshal, 574
 - clearBody, 574
 - clone, 574
 - cloneDataStructure, 575
 - copyDataStructure, 575
 - equals, 575
 - getDataStructureType, 575
 - getSize, 575
 - getText, 576
 - ID_ACTIVEMQTEXTMESSAGE, 577
 - setText, 576
 - text, 577
 - toString, 577
- activemq::commands::ActiveMQTopic, 598
 - ~ActiveMQTopic, 599
 - ActiveMQTopic, 599
 - clone, 599
 - cloneDataStructure, 599
 - copy, 599
 - copyDataStructure, 599
 - equals, 600
 - getCMSDestination, 600
 - getCMSProperties, 600
 - getDataStructureType, 600
 - getDestinationType, 600
 - getTopicName, 601
 - ID_ACTIVEMQTOPIC, 601
 - toString, 601
- activemq::commands::BaseCommand, 650
 - ~BaseCommand, 651
 - BaseCommand, 651
 - copyDataStructure, 651
 - equals, 651
 - getCommandId, 652
 - isBrokerInfo, 652
 - isConnectionInfo, 653
 - isConsumerInfo, 653
 - isKeepAliveInfo, 653
 - isMessage, 653
 - isMessageAck, 653
 - isMessageDispatch, 653
 - isMessageDispatchNotification, 653
 - isProducerAck, 654
 - isProducerInfo, 654
 - isRemoveInfo, 654
 - isRemoveSubscriptionInfo, 654
 - isResponse, 654
 - isResponseRequired, 654
 - isShutdownInfo, 655
 - isTransactionInfo, 655

- isWireFormatInfo, 655
- setCommandId, 655
- setResponseRequired, 655
- toString, 655
- activemq::commands::BaseDataStructure, 715
 - ~BaseDataStructure, 716
 - afterMarshal, 716
 - afterUnmarshal, 716
 - beforeMarshal, 716
 - beforeUnmarshal, 716
 - copyDataStructure, 716
 - equals, 717
 - getMarshaledForm, 717
 - isMarshalAware, 717
 - setMarshaledForm, 717
 - toString, 718
- activemq::commands::BooleanExpression, 737
 - ~BooleanExpression, 737
 - BooleanExpression, 737
 - cloneDataStructure, 737
 - copyDataStructure, 737
 - equals, 738
 - toString, 738
- activemq::commands::BrokerError, 745
 - ~BrokerError, 746
 - BrokerError, 746
 - cloneDataStructure, 746
 - copyDataStructure, 746
 - getCause, 746
 - getDataStructureType, 747
 - getExceptionClass, 747
 - getMessage, 747
 - getStackTraceElements, 747
 - setCause, 747
 - setExceptionClass, 748
 - setMessage, 748
 - setStackTraceElements, 748
 - visit, 748
- activemq::commands::BrokerError::StackTraceElement
 - 2993
 - ClassName, 2993
 - FileName, 2993
 - LineNumber, 2993
 - MethodName, 2993
- activemq::commands::BrokerId, 751
 - ~BrokerId, 752
 - BrokerId, 752
 - cloneDataStructure, 752
 - COMPARATOR, 752
 - compareTo, 752
 - copyDataStructure, 752
 - equals, 752
 - getDataStructureType, 753
 - getValue, 753
 - ID_BROKERID, 753
 - operator<, 753
 - operator=, 753
 - operator==, 753
 - setValue, 753
 - toString, 753
 - value, 753
- activemq::commands::BrokerInfo, 775
 - ~BrokerInfo, 777
 - brokerId, 782
 - BrokerInfo, 777
 - brokerName, 782
 - brokerUploadUrl, 782
 - brokerURL, 782
 - cloneDataStructure, 777
 - connectionId, 782
 - copyDataStructure, 777
 - duplexConnection, 782
 - equals, 777
 - faultTolerantConfiguration, 782
 - getBrokerId, 777, 778
 - getBrokerName, 778
 - getBrokerUploadUrl, 778
 - getBrokerURL, 778
 - getConnectionId, 778
 - getDataStructureType, 778
 - getNetworkProperties, 778, 779
 - getPeerBrokerInfos, 779
 - ID_BROKERINFO, 782
 - isBrokerInfo, 779
 - isDuplexConnection, 779
 - isFaultTolerantConfiguration, 780
 - isMasterBroker, 780
 - isNetworkConnection, 780
 - isSlaveBroker, 780
 - masterBroker, 782
 - networkConnection, 782
 - networkProperties, 782
 - operator=, 780
 - peerBrokerInfos, 782
 - setBrokerId, 780
 - setBrokerName, 780
 - setBrokerUploadUrl, 780
 - setBrokerURL, 780
 - setConnectionId, 780
 - setDuplexConnection, 780
 - setFaultTolerantConfiguration, 780
 - setMasterBroker, 780
 - setNetworkConnection, 780
 - setNetworkProperties, 780
 - setPeerBrokerInfos, 780
 - setSlaveBroker, 780
 - slaveBroker, 782
 - toString, 780

- visit, 781
- activemq::commands::Command, 1063
 - ~Command, 1064
 - getCommandId, 1064
 - isBrokerInfo, 1064
 - isConnectionInfo, 1064
 - isConsumerInfo, 1064
 - isKeepAliveInfo, 1064
 - isMessage, 1064
 - isMessageAck, 1064
 - isMessageDispatch, 1065
 - isMessageDispatchNotification, 1065
 - isProducerAck, 1065
 - isProducerInfo, 1065
 - isRemoveInfo, 1065
 - isRemoveSubscriptionInfo, 1065
 - isResponse, 1065
 - isResponseRequired, 1065
 - isShutdownInfo, 1066
 - isTransactionInfo, 1066
 - isWireFormatInfo, 1066
 - setCommandId, 1066
 - setResponseRequired, 1066
 - toString, 1066
 - visit, 1067
- activemq::commands::ConnectionControl, 1135
 - ~ConnectionControl, 1136
 - cloneDataStructure, 1136
 - close, 1139
 - ConnectionControl, 1136
 - copyDataStructure, 1136
 - equals, 1136
 - exit, 1139
 - faultTolerant, 1139
 - getDataStructureType, 1137
 - ID_CONNECTIONCONTROL, 1139
 - isClose, 1137
 - isExit, 1138
 - isFaultTolerant, 1138
 - isResume, 1138
 - isSuspend, 1138
 - operator=, 1138
 - resume, 1139
 - setClose, 1138
 - setExit, 1138
 - setFaultTolerant, 1138
 - setResume, 1138
 - setSuspend, 1138
 - suspend, 1139
 - toString, 1138
 - visit, 1138
- activemq::commands::ConnectionError, 1160
 - ~ConnectionError, 1161
 - cloneDataStructure, 1161
 - ConnectionError, 1161
 - connectionId, 1163
 - copyDataStructure, 1161
 - equals, 1161
 - exception, 1163
 - getConnectionId, 1161, 1162
 - getDataStructureType, 1162
 - getException, 1162
 - ID_CONNECTIONERROR, 1163
 - operator=, 1162
 - setConnectionId, 1162
 - setException, 1162
 - toString, 1162
 - visit, 1162
- activemq::commands::ConnectionId, 1187
 - ~ConnectionId, 1188
 - cloneDataStructure, 1188
 - COMPARATOR, 1188
 - compareTo, 1188
 - ConnectionId, 1188
 - copyDataStructure, 1188
 - equals, 1189
 - getDataStructureType, 1189
 - getValue, 1189
 - ID_CONNECTIONID, 1190
 - operator<, 1189
 - operator=, 1189
 - operator==, 1189
 - setValue, 1189
 - toString, 1189
 - value, 1190
- activemq::commands::ConnectionInfo, 1211
 - ~ConnectionInfo, 1212
 - brokerMasterConnector, 1216
 - brokerPath, 1216
 - clientId, 1216
 - clientMaster, 1216
 - cloneDataStructure, 1212
 - connectionId, 1216
 - ConnectionInfo, 1212
 - copyDataStructure, 1212
 - equals, 1213
 - getBrokerPath, 1213
 - getClientId, 1213
 - getConnectionId, 1213
 - getDataStructureType, 1213
 - getPassword, 1213, 1214
 - getUserName, 1214
 - ID_CONNECTIONINFO, 1216
 - isBrokerMasterConnector, 1214
 - isClientMaster, 1214
 - isConnectionInfo, 1214
 - isManageable, 1214
 - manageable, 1216

- operator=, 1215
- password, 1216
- setBrokerMasterConnector, 1215
- setBrokerPath, 1215
- setClientId, 1215
- setClientMaster, 1215
- setConnectionId, 1215
- setManageable, 1215
- setPassword, 1215
- setUserName, 1215
- toString, 1215
- userName, 1216
- visit, 1215
- activemq::commands::ConsumerControl, 1250
 - ~ConsumerControl, 1251
 - cloneDataStructure, 1251
 - close, 1254
 - ConsumerControl, 1251
 - consumerId, 1254
 - copyDataStructure, 1251
 - equals, 1251
 - flush, 1254
 - getConsumerId, 1252
 - getDataStructureType, 1252
 - getPrefetch, 1252
 - ID_CONSUMERCONTROL, 1254
 - isClose, 1253
 - isFlush, 1253
 - isStart, 1253
 - isStop, 1253
 - operator=, 1253
 - prefetch, 1254
 - setClose, 1253
 - setConsumerId, 1253
 - setFlush, 1253
 - setPrefetch, 1253
 - setStart, 1253
 - setStop, 1253
 - start, 1254
 - stop, 1254
 - toString, 1253
 - visit, 1253
- activemq::commands::ConsumerId, 1275
 - ~ConsumerId, 1276
 - cloneDataStructure, 1276
 - COMPARATOR, 1276
 - compareTo, 1276
 - connectionId, 1278
 - ConsumerId, 1276
 - copyDataStructure, 1276
 - equals, 1277
 - getConnectionId, 1277
 - getDataStructureType, 1277
 - getParentId, 1277
 - getSessionId, 1278
 - getValue, 1278
 - ID_CONSUMERID, 1278
 - operator<, 1278
 - operator=, 1278
 - operator==, 1278
 - sessionId, 1278
 - setConnectionId, 1278
 - setSessionId, 1278
 - setValue, 1278
 - toString, 1278
 - value, 1279
- activemq::commands::ConsumerInfo, 1300
 - ~ConsumerInfo, 1302
 - additionalPredicate, 1307
 - brokerPath, 1307
 - browser, 1307
 - cloneDataStructure, 1302
 - consumerId, 1307
 - ConsumerInfo, 1302
 - copyDataStructure, 1302
 - destination, 1307
 - dispatchAsync, 1307
 - equals, 1302
 - exclusive, 1307
 - getAdditionalPredicate, 1303
 - getBrokerPath, 1303
 - getConsumerId, 1303
 - getDataStructureType, 1303
 - getDestination, 1303, 1304
 - getMaximumPendingMessageLimit, 1304
 - getNetworkConsumerPath, 1304
 - getPrefetchSize, 1304
 - getPriority, 1304
 - getSelector, 1304
 - getSubscriptionName, 1304
 - ID_CONSUMERINFO, 1307
 - isBrowser, 1304
 - isConsumerInfo, 1304
 - isDispatchAsync, 1304
 - isExclusive, 1305
 - isNetworkSubscription, 1305
 - isNoLocal, 1305
 - isNoRangeAcks, 1305
 - isOptimizedAcknowledge, 1305
 - isRetroactive, 1305
 - maximumPendingMessageLimit, 1307
 - networkConsumerPath, 1307
 - networkSubscription, 1307
 - noLocal, 1307
 - noRangeAcks, 1307
 - operator=, 1305
 - optimizedAcknowledge, 1307
 - prefetchSize, 1307

- priority, 1307
- retroactive, 1307
- selector, 1307
- setAdditionalPredicate, 1305
- setBrokerPath, 1305
- setBrowser, 1305
- setConsumerId, 1305
- setDestination, 1305
- setDispatchAsync, 1305
- setExclusive, 1305
- setMaximumPendingMessageLimit, 1305
- setNetworkConsumerPath, 1305
- setNetworkSubscription, 1305
- setNoLocal, 1305
- setNoRangeAcks, 1305
- setOptimizedAcknowledge, 1305
- setPrefetchSize, 1305
- setPriority, 1305
- setRetroactive, 1305
- setSelector, 1305
- setSubscriptionName, 1305
- subscriptionName, 1307
- toString, 1305
- visit, 1306
- activemq::commands::ControlCommand, 1330
 - ~ControlCommand, 1331
 - cloneDataStructure, 1331
 - command, 1333
 - ControlCommand, 1331
 - copyDataStructure, 1331
 - equals, 1331
 - getCommand, 1331, 1332
 - getDataStructureType, 1332
 - ID_CONTROLCOMMAND, 1333
 - operator=, 1332
 - setCommand, 1332
 - toString, 1332
 - visit, 1332
- activemq::commands::DataArrayResponse, 1356
 - ~DataArrayResponse, 1357
 - cloneDataStructure, 1357
 - copyDataStructure, 1357
 - data, 1358
 - DataArrayResponse, 1357
 - equals, 1357
 - getData, 1357, 1358
 - getDataStructureType, 1358
 - ID_DATAARRAYRESPONSE, 1358
 - operator=, 1358
 - setData, 1358
 - toString, 1358
- activemq::commands::DataResponse, 1395
 - ~DataResponse, 1396
 - cloneDataStructure, 1396
 - copyDataStructure, 1396
 - data, 1397
 - DataResponse, 1396
 - equals, 1396
 - getData, 1396, 1397
 - getDataStructureType, 1397
 - ID_DATARESPONSE, 1397
 - operator=, 1397
 - setData, 1397
 - toString, 1397
- activemq::commands::DataStructure, 1461
 - ~DataStructure, 1461
 - cloneDataStructure, 1461
 - copyDataStructure, 1462
 - equals, 1463
 - getDataStructureType, 1464
 - toString, 1465
- activemq::commands::DestinationInfo, 1484
 - ~DestinationInfo, 1485
 - brokerPath, 1488
 - cloneDataStructure, 1485
 - connectionId, 1488
 - copyDataStructure, 1485
 - destination, 1488
 - DestinationInfo, 1485
 - equals, 1485
 - getBrokerPath, 1486
 - getConnectionId, 1486
 - getDataStructureType, 1486
 - getDestination, 1486, 1487
 - getOperationType, 1487
 - getTimeout, 1487
 - ID_DESTINATIONINFO, 1488
 - operationType, 1488
 - operator=, 1487
 - setBrokerPath, 1487
 - setConnectionId, 1487
 - setDestination, 1487
 - setOperationType, 1487
 - setTimeout, 1487
 - timeout, 1488
 - toString, 1487
 - visit, 1487
- activemq::commands::DiscoveryEvent, 1511
 - ~DiscoveryEvent, 1512
 - brokerName, 1514
 - cloneDataStructure, 1512
 - copyDataStructure, 1512
 - DiscoveryEvent, 1512
 - equals, 1512
 - getBrokerName, 1512, 1513
 - getDataStructureType, 1513
 - getServiceName, 1513

- ID_DISCOVERYEVENT, 1514
- operator=, 1513
- serviceName, 1514
- setBrokerName, 1513
- setServiceName, 1513
- toString, 1513
- activemq::commands::ExceptionResponse, 1582
 - ~ExceptionResponse, 1583
 - cloneDataStructure, 1583
 - copyDataStructure, 1583
 - equals, 1583
 - exception, 1584
 - ExceptionResponse, 1583
 - getDataStructureType, 1583
 - getException, 1584
 - ID_EXCEPTIONRESPONSE, 1584
 - operator=, 1584
 - setException, 1584
 - toString, 1584
- activemq::commands::FlushCommand, 1676
 - ~FlushCommand, 1677
 - cloneDataStructure, 1677
 - copyDataStructure, 1677
 - equals, 1677
 - FlushCommand, 1677
 - getDataStructureType, 1677
 - ID_FLUSHCOMMAND, 1678
 - operator=, 1678
 - toString, 1678
 - visit, 1678
- activemq::commands::IntegerResponse, 1777
 - ~IntegerResponse, 1778
 - cloneDataStructure, 1778
 - copyDataStructure, 1778
 - equals, 1778
 - getDataStructureType, 1778
 - getResult, 1779
 - ID_INTEGERRESPONSE, 1779
 - IntegerResponse, 1778
 - operator=, 1779
 - result, 1779
 - setResult, 1779
 - toString, 1779
- activemq::commands::JournalQueueAck, 1834
 - ~JournalQueueAck, 1835
 - cloneDataStructure, 1835
 - copyDataStructure, 1835
 - destination, 1837
 - equals, 1835
 - getDataStructureType, 1835
 - getDestination, 1836
 - getMessageAck, 1836
 - ID_JOURNALQUEUEACK, 1837
 - JournalQueueAck, 1835
 - messageAck, 1837
 - operator=, 1836
 - setDestination, 1836
 - setMessageAck, 1836
 - toString, 1836
- activemq::commands::JournalTopicAck, 1858
 - ~JournalTopicAck, 1859
 - clientId, 1862
 - cloneDataStructure, 1859
 - copyDataStructure, 1859
 - destination, 1862
 - equals, 1859
 - getClientId, 1860
 - getDataStructureType, 1860
 - getDestination, 1860, 1861
 - getMessageId, 1861
 - getMessageSequenceId, 1861
 - getSubscriptionName, 1861
 - getTransactionId, 1861
 - ID_JOURNALTOPICACK, 1862
 - JournalTopicAck, 1859
 - messageId, 1862
 - messageSequenceId, 1862
 - operator=, 1861
 - setClientId, 1861
 - setDestination, 1861
 - setMessageId, 1861
 - setMessageSequenceId, 1861
 - setSubscriptionName, 1861
 - setTransactionId, 1861
 - subscriptionName, 1862
 - toString, 1861
 - transactionId, 1862
- activemq::commands::JournalTrace, 1883
 - ~JournalTrace, 1884
 - cloneDataStructure, 1884
 - copyDataStructure, 1884
 - equals, 1884
 - getDataStructureType, 1884
 - getMessage, 1885
 - ID_JOURNALTRACE, 1885
 - JournalTrace, 1884
 - message, 1885
 - operator=, 1885
 - setMessage, 1885
 - toString, 1885
- activemq::commands::JournalTransaction, 1906
 - ~JournalTransaction, 1907
 - cloneDataStructure, 1907
 - copyDataStructure, 1907
 - equals, 1907
 - getDataStructureType, 1907
 - getTransactionId, 1908
 - getType, 1908

- getWasPrepared, 1908
- ID_ JOURNALTRANSACTION, 1909
- JournalTransaction, 1907
- operator=, 1908
- setTransactionId, 1908
- setType, 1908
- setWasPrepared, 1908
- toString, 1908
- transactionId, 1909
- type, 1909
- wasPrepared, 1909
- activemq::commands::KeepAliveInfo, 1930
 - ~KeepAliveInfo, 1931
 - cloneDataStructure, 1931
 - copyDataStructure, 1931
 - equals, 1931
 - getDataStructureType, 1931
 - ID_ KEEPALIVEINFO, 1932
 - isKeepAliveInfo, 1932
 - KeepAliveInfo, 1931
 - operator=, 1932
 - toString, 1932
 - visit, 1932
- activemq::commands::LastPartialCommand, 1958
 - ~LastPartialCommand, 1959
 - cloneDataStructure, 1959
 - copyDataStructure, 1959
 - equals, 1959
 - getDataStructureType, 1959
 - ID_ LASTPARTIALCOMMAND, 1960
 - LastPartialCommand, 1959
 - operator=, 1960
 - toString, 1960
- activemq::commands::LocalTransactionId, 1993
 - ~LocalTransactionId, 1994
 - cloneDataStructure, 1994
 - COMPARATOR, 1994
 - compareTo, 1994
 - connectionId, 1996
 - copyDataStructure, 1994
 - equals, 1994, 1995
 - getConnectionId, 1995
 - getDataStructureType, 1995
 - getValue, 1995
 - ID_ LOCALTRANSACTIONID, 1996
 - LocalTransactionId, 1994
 - operator<, 1996
 - operator=, 1996
 - operator==, 1996
 - setConnectionId, 1996
 - setValue, 1996
 - toString, 1996
 - value, 1996
- activemq::commands::Message, 2145
 - ~Message, 2149
 - afterUnmarshal, 2149
 - arrival, 2161
 - beforeMarshal, 2149
 - brokerInTime, 2161
 - brokerOutTime, 2161
 - brokerPath, 2161
 - cloneDataStructure, 2149
 - cluster, 2161
 - compressed, 2161
 - content, 2161
 - copyDataStructure, 2150
 - correlationId, 2161
 - dataStructure, 2161
 - DEFAULT_ MESSAGE_ SIZE, 2161
 - destination, 2161
 - droppable, 2161
 - equals, 2150
 - expiration, 2161
 - getAckHandler, 2150
 - getArrival, 2151
 - getBrokerInTime, 2152
 - getBrokerOutTime, 2152
 - getBrokerPath, 2152
 - getCluster, 2152
 - getContent, 2152
 - getCorrelationId, 2152
 - getDataStructure, 2152
 - getDataStructureType, 2152
 - getDestination, 2153
 - getExpiration, 2153
 - getGroupID, 2153
 - getGroupSequence, 2153
 - getMarshaledProperties, 2153
 - getMessageId, 2153
 - getMessageProperties, 2153
 - getOriginalDestination, 2154
 - getOriginalTransactionId, 2154
 - getPriority, 2154
 - getProducerId, 2154
 - getRedeliveryCounter, 2154
 - getReplyTo, 2154
 - getSize, 2154
 - getTargetConsumerId, 2154, 2155
 - getTimestamp, 2155
 - getTransactionId, 2155
 - getType, 2155
 - getUserID, 2155
 - groupID, 2161
 - groupSequence, 2161
 - ID_ MESSAGE, 2161
 - isCompressed, 2155
 - isDroppable, 2155

- isExpired, 2155
- isMarshalAware, 2155
- isMessage, 2156
- isPersistent, 2156
- isReadOnlyBody, 2156
- isReadOnlyProperties, 2156
- isRecievedByDFBridge, 2156
- marshalledProperties, 2161
- Message, 2149
- messageId, 2161
- onSend, 2156
- operator=, 2156
- originalDestination, 2161
- originalTransactionId, 2161
- persistent, 2161
- priority, 2161
- producerId, 2161
- recievedByDFBridge, 2161
- redeliveryCounter, 2161
- replyTo, 2161
- setAckHandler, 2157
- setArrival, 2157
- setBrokerInTime, 2158
- setBrokerOutTime, 2158
- setBrokerPath, 2158
- setCluster, 2158
- setCompressed, 2158
- setContent, 2158
- setCorrelationId, 2158
- setDataStructure, 2158
- setDestination, 2158
- setDroppable, 2158
- setExpiration, 2158
- setGroupID, 2158
- setGroupSequence, 2158
- setMarshalledProperties, 2158
- setMessageId, 2158
- setOriginalDestination, 2158
- setOriginalTransactionId, 2158
- setPersistent, 2158
- setPriority, 2158
- setProducerId, 2158
- setReadOnlyBody, 2158
- setReadOnlyProperties, 2159
- setRecievedByDFBridge, 2159
- setRedeliveryCounter, 2159
- setReplyTo, 2159
- setTargetConsumerId, 2159
- setTimestamp, 2159
- setTransactionId, 2159
- setType, 2159
- setUserID, 2159
- targetConsumerId, 2161
- timestamp, 2161
- toString, 2159
- transactionId, 2161
- type, 2161
- userID, 2161
- visit, 2159
- activemq::commands::MessageAck, 2188
 - ~MessageAck, 2189
 - ackType, 2193
 - cloneDataStructure, 2189
 - consumerId, 2193
 - copyDataStructure, 2189
 - destination, 2193
 - equals, 2189
 - firstMessageId, 2193
 - getAckType, 2190
 - getConsumerId, 2190
 - getDataStructureType, 2190
 - getDestination, 2190, 2191
 - getFirstMessageId, 2191
 - getLastMessageId, 2191
 - getMessageCount, 2191
 - getTransactionId, 2191
 - ID_MESSAGEACK, 2193
 - isMessageAck, 2191
 - lastMessageId, 2193
 - MessageAck, 2189
 - messageCount, 2193
 - operator=, 2191
 - setAckType, 2192
 - setConsumerId, 2192
 - setDestination, 2192
 - setFirstMessageId, 2192
 - setLastMessageId, 2192
 - setMessageCount, 2192
 - setTransactionId, 2192
 - toString, 2192
 - transactionId, 2193
 - visit, 2192
- activemq::commands::MessageDispatch, 2219
 - ~MessageDispatch, 2220
 - cloneDataStructure, 2220
 - consumerId, 2223
 - copyDataStructure, 2220
 - destination, 2223
 - equals, 2220
 - getConsumerId, 2221
 - getDataStructureType, 2221
 - getDestination, 2221, 2222
 - getMessage, 2222
 - getRedeliveryCounter, 2222
 - ID_MESSAGEDISPATCH, 2223
 - isMessageDispatch, 2222
 - message, 2223
 - MessageDispatch, 2220

- operator=, 2222
- redeliveryCounter, 2223
- setConsumerId, 2222
- setDestination, 2222
- setMessage, 2222
- setRedeliveryCounter, 2222
- toString, 2222
- visit, 2223
- activemq::commands::MessageDispatchNotification, 2251
 - ~MessageDispatchNotification, 2252
 - cloneDataStructure, 2252
 - consumerId, 2256
 - copyDataStructure, 2252
 - deliverySequenceId, 2256
 - destination, 2256
 - equals, 2252
 - getConsumerId, 2253
 - getDataStructureType, 2253
 - getDeliverySequenceId, 2253
 - getDestination, 2254
 - getMessageId, 2254
 - ID_MESSAGEDISPATCHNOTIFICATION, 2256
 - isMessageDispatchNotification, 2254
 - MessageDispatchNotification, 2252
 - messageId, 2256
 - operator=, 2254
 - setConsumerId, 2255
 - setDeliverySequenceId, 2255
 - setDestination, 2255
 - setMessageId, 2255
 - toString, 2255
 - visit, 2255
- activemq::commands::MessageId, 2279
 - ~MessageId, 2280
 - brokerSequenceId, 2283
 - cloneDataStructure, 2280
 - COMPARATOR, 2280
 - compareTo, 2280
 - copyDataStructure, 2280
 - equals, 2280, 2281
 - getBrokerSequenceId, 2281
 - getDataStructureType, 2281
 - getProducerId, 2281, 2282
 - getProducerSequenceId, 2282
 - ID_MESSAGEID, 2283
 - MessageId, 2280
 - operator<, 2282
 - operator=, 2282
 - operator==, 2282
 - producerId, 2283
 - producerSequenceId, 2283
 - setBrokerSequenceId, 2282
 - setProducerId, 2282
 - setProducerSequenceId, 2282
 - toString, 2282
 - visit, 2282
- activemq::commands::MessagePull, 2346
 - ~MessagePull, 2347
 - cloneDataStructure, 2347
 - consumerId, 2350
 - copyDataStructure, 2347
 - correlationId, 2350
 - destination, 2350
 - equals, 2347
 - getConsumerId, 2348
 - getCorrelationId, 2348
 - getDataStructureType, 2348
 - getDestination, 2348, 2349
 - getMessageId, 2349
 - getTimeout, 2349
 - ID_MESSAGEPULL, 2350
 - messageId, 2350
 - MessagePull, 2347
 - operator=, 2349
 - setConsumerId, 2349
 - setCorrelationId, 2349
 - setDestination, 2349
 - setMessageId, 2349
 - setTimeout, 2349
 - timeout, 2350
 - toString, 2349
 - visit, 2349
- activemq::commands::NetworkBridgeFilter, 2393
 - ~NetworkBridgeFilter, 2394
 - cloneDataStructure, 2394
 - copyDataStructure, 2394
 - equals, 2394
 - getDataStructureType, 2394
 - getNetworkBrokerId, 2395
 - getNetworkTTL, 2395
 - ID_NETWORKBRIDGEFILTER, 2396
 - NetworkBridgeFilter, 2394
 - networkBrokerId, 2396
 - networkTTL, 2396
 - operator=, 2395
 - setNetworkBrokerId, 2395
 - setNetworkTTL, 2395
 - toString, 2395
- activemq::commands::PartialCommand, 2473
 - ~PartialCommand, 2474
 - cloneDataStructure, 2474
 - commandId, 2476
 - copyDataStructure, 2474
 - data, 2476
 - equals, 2474
 - getCommandId, 2474

- getData, 2475
 - getDataStructureType, 2475
 - ID_PARTIALCOMMAND, 2476
 - operator=, 2475
 - PartialCommand, 2474
 - setCommandId, 2475
 - setData, 2475
 - toString, 2475
- activemq::commands::ProducerAck, 2579
 - ~ProducerAck, 2580
 - cloneDataStructure, 2580
 - copyDataStructure, 2580
 - equals, 2580
 - getDataStructureType, 2580
 - getProducerId, 2581
 - getSize, 2581
 - ID_PRODUCERACK, 2582
 - isProducerAck, 2581
 - operator=, 2581
 - ProducerAck, 2580
 - producerId, 2582
 - setProducerId, 2581
 - setSize, 2581
 - size, 2582
 - toString, 2581
 - visit, 2581
- activemq::commands::ProducerId, 2606
 - ~ProducerId, 2607
 - cloneDataStructure, 2607
 - COMPARATOR, 2607
 - compareTo, 2607
 - connectionId, 2609
 - copyDataStructure, 2607
 - equals, 2608
 - getConnectionId, 2608
 - getDataStructureType, 2608
 - getParentId, 2608
 - getSessionId, 2609
 - getValue, 2609
 - ID_PRODUCERID, 2609
 - operator<, 2609
 - operator=, 2609
 - operator==, 2609
 - ProducerId, 2607
 - sessionId, 2609
 - setConnectionId, 2609
 - setSessionId, 2609
 - setValue, 2609
 - toString, 2609
 - value, 2610
- activemq::commands::ProducerInfo, 2631
 - ~ProducerInfo, 2632
 - brokerPath, 2635
 - cloneDataStructure, 2632
 - copyDataStructure, 2632
 - destination, 2635
 - dispatchAsync, 2635
 - equals, 2632
 - getBrokerPath, 2633
 - getDataStructureType, 2633
 - getDestination, 2633, 2634
 - getProducerId, 2634
 - getWindowSize, 2634
 - ID_PRODUCERINFO, 2635
 - isDispatchAsync, 2634
 - isProducerInfo, 2634
 - operator=, 2634
 - producerId, 2635
 - ProducerInfo, 2632
 - setBrokerPath, 2634
 - setDestination, 2634
 - setDispatchAsync, 2634
 - setProducerId, 2634
 - setWindowSize, 2634
 - toString, 2634
 - visit, 2635
 - windowSize, 2635
- activemq::commands::RemoveInfo, 2703
 - ~RemoveInfo, 2704
 - cloneDataStructure, 2704
 - copyDataStructure, 2704
 - equals, 2704
 - getDataStructureType, 2704
 - getLastDeliveredSequenceId, 2705
 - getObjectId, 2705
 - ID_REMOVEINFO, 2706
 - isRemoveInfo, 2705
 - lastDeliveredSequenceId, 2706
 - objectId, 2706
 - operator=, 2705
 - RemoveInfo, 2704
 - setLastDeliveredSequenceId, 2705
 - setObjectId, 2705
 - toString, 2705
 - visit, 2705
- activemq::commands::RemoveSubscriptionInfo, 2727
 - ~RemoveSubscriptionInfo, 2728
 - clientId, 2731
 - cloneDataStructure, 2728
 - connectionId, 2731
 - copyDataStructure, 2728
 - equals, 2728
 - getClientId, 2729
 - getConnectionId, 2729
 - getDataStructureType, 2729
 - getSubscriptionName, 2729, 2730

- ID_REMOVE SUBSCRIPTIONINFO, 2731
- isRemoveSubscriptionInfo, 2730
- operator=, 2730
- RemoveSubscriptionInfo, 2728
- setClientId, 2730
- setConnectionId, 2730
- setSubscriptionName, 2730
- subscriptionName, 2731
- toString, 2730
- visit, 2730
- activemq::commands::ReplayCommand, 2752
 - ~ReplayCommand, 2753
 - cloneDataStructure, 2753
 - copyDataStructure, 2753
 - equals, 2753
 - firstNakNumber, 2755
 - getDataStructureType, 2753
 - getFirstNakNumber, 2754
 - getLastNakNumber, 2754
 - ID_REPLAYCOMMAND, 2755
 - lastNakNumber, 2755
 - operator=, 2754
 - ReplayCommand, 2753
 - setFirstNakNumber, 2754
 - setLastNakNumber, 2754
 - toString, 2754
 - visit, 2754
- activemq::commands::Response, 2781
 - ~Response, 2782
 - cloneDataStructure, 2782
 - copyDataStructure, 2782
 - correlationId, 2784
 - equals, 2782
 - getCorrelationId, 2783
 - getDataStructureType, 2783
 - ID_RESPONSE, 2784
 - isResponse, 2783
 - operator=, 2783
 - Response, 2782
 - setCorrelationId, 2783
 - toString, 2783
 - visit, 2783
- activemq::commands::SessionId, 2853
 - ~SessionId, 2854
 - cloneDataStructure, 2854
 - COMPARATOR, 2854
 - compareTo, 2854
 - connectionId, 2856
 - copyDataStructure, 2854
 - equals, 2855
 - getConnectionId, 2855
 - getDataStructureType, 2855
 - getParentId, 2855
 - getValue, 2855
 - ID_SESSIONID, 2856
 - operator<, 2855
 - operator=, 2856
 - operator==, 2856
 - SessionId, 2854
 - setConnectionId, 2856
 - setValue, 2856
 - toString, 2856
 - value, 2856
- activemq::commands::SessionInfo, 2877
 - ~SessionInfo, 2878
 - cloneDataStructure, 2878
 - copyDataStructure, 2878
 - equals, 2878
 - getAckMode, 2878
 - getDataStructureType, 2879
 - getSessionId, 2879
 - ID_SESSIONINFO, 2880
 - operator=, 2879
 - sessionId, 2880
 - SessionInfo, 2878
 - setAckMode, 2879
 - setSessionId, 2879
 - toString, 2879
 - visit, 2879
- activemq::commands::ShutdownInfo, 2934
 - ~ShutdownInfo, 2935
 - cloneDataStructure, 2935
 - copyDataStructure, 2935
 - equals, 2935
 - getDataStructureType, 2935
 - ID_SHUTDOWNINFO, 2936
 - isShutdownInfo, 2936
 - operator=, 2936
 - ShutdownInfo, 2935
 - toString, 2936
 - visit, 2936
- activemq::commands::SubscriptionInfo, 3097
 - ~SubscriptionInfo, 3098
 - clientId, 3101
 - cloneDataStructure, 3098
 - copyDataStructure, 3098
 - destination, 3101
 - equals, 3098
 - getClientId, 3099
 - getDataStructureType, 3099
 - getDestination, 3099, 3100
 - getSelector, 3100
 - getSubscriptionName, 3100
 - getSubscribedDestination, 3100
 - ID_SUBSCRIPTIONINFO, 3101
 - operator=, 3100
 - selector, 3101

- setClientId, 3100
 - setDestination, 3100
 - setSelector, 3100
 - setSubscriptionName, 3100
 - setSubscribedDestination, 3100
 - subscriptionName, 3101
 - subscribedDestination, 3101
 - SubscriptionInfo, 3098
 - toString, 3100
- activemq::commands::TransactionId, 3217
 - ~TransactionId, 3218
 - cloneDataStructure, 3218
 - COMPARATOR, 3217
 - compareTo, 3218
 - copyDataStructure, 3218
 - equals, 3218
 - getDataStructureType, 3219
 - ID_TRANSACTIONID, 3219
 - operator<, 3219
 - operator=, 3219
 - operator==, 3219
 - toString, 3219
 - TransactionId, 3218
- activemq::commands::TransactionInfo, 3240
 - ~TransactionInfo, 3241
 - cloneDataStructure, 3241
 - connectionId, 3243
 - copyDataStructure, 3241
 - equals, 3241
 - getConnectionId, 3241, 3242
 - getDataStructureType, 3242
 - getTransactionId, 3242
 - getType, 3242
 - ID_TRANSACTIONINFO, 3243
 - isTransactionInfo, 3242
 - operator=, 3242
 - setConnectionId, 3243
 - setTransactionId, 3243
 - setType, 3243
 - toString, 3243
 - transactionId, 3243
 - TransactionInfo, 3241
 - type, 3243
 - visit, 3243
- activemq::commands::WireFormatInfo, 3369
 - ~WireFormatInfo, 3371
 - afterUnmarshal, 3371
 - beforeMarshal, 3372
 - cloneDataStructure, 3372
 - copyDataStructure, 3372
 - equals, 3372
 - getCacheSize, 3372
 - getDataStructureType, 3373
 - getMagic, 3373
 - getMarshaledProperties, 3373
 - getMaxInactivityDuration, 3373
 - getMaxInactivityDurationInitialDelay, 3373
 - getProperties, 3374
 - getVersion, 3374
 - ID_WIREFORMATINFO, 3378
 - isCacheEnabled, 3374
 - isMarshalAware, 3374
 - isSizePrefixDisabled, 3375
 - isStackTraceEnabled, 3375
 - isTcpNoDelayEnabled, 3375
 - isTightEncodingEnabled, 3375
 - isValid, 3375
 - isWireFormatInfo, 3375
 - setCacheEnabled, 3376
 - setCacheSize, 3376
 - setMagic, 3376
 - setMarshaledProperties, 3376
 - setMaxInactivityDuration, 3376
 - setMaxInactivityDurationInitialDelay, 3376
 - setProperties, 3377
 - setSizePrefixDisabled, 3377
 - setStackTraceEnabled, 3377
 - setTcpNoDelayEnabled, 3377
 - setTightEncodingEnabled, 3377
 - setVersion, 3377
 - toString, 3378
 - visit, 3378
 - WireFormatInfo, 3371
- activemq::commands::XATransactionId, 3410
 - ~XATransactionId, 3411
 - branchQualifier, 3414
 - cloneDataStructure, 3411
 - COMPARATOR, 3411
 - compareTo, 3411
 - copyDataStructure, 3411
 - equals, 3412
 - formatId, 3414
 - getBranchQualifier, 3412
 - getDataStructureType, 3412
 - getFormatId, 3412
 - getGlobalTransactionId, 3413
 - globalTransactionId, 3414
 - ID_XATRANSACTIONID, 3414
 - operator<, 3413
 - operator=, 3413
 - operator==, 3413
 - setBranchQualifier, 3413
 - setFormatId, 3413
 - setGlobalTransactionId, 3413
 - toString, 3413
 - XATransactionId, 3411
- activemq::core, 73
- activemq::core::ActiveMQAckHandler, 171

- ~ActiveMQAckHandler, 171
- acknowledgeMessage, 171
- activemq::core::ActiveMQConnection, 233
 - ~ActiveMQConnection, 235
 - ActiveMQConnection, 235
 - addDispatcher, 236
 - addProducer, 236
 - addTransportListener, 236
 - close, 236
 - createSession, 236, 237
 - destroyDestination, 237
 - fire, 238
 - getClientID, 238
 - getConnectionId, 238
 - getConnectionInfo, 238
 - getExceptionListener, 239
 - getMetaData, 239
 - isClosed, 239
 - isStarted, 239
 - onCommand, 240
 - oneway, 240
 - onException, 240
 - removeDispatcher, 240
 - removeProducer, 240
 - removeSession, 241
 - removeTransportListener, 241
 - sendPullRequest, 241
 - setExceptionListener, 241
 - start, 241
 - stop, 242
 - syncRequest, 242
 - transportInterrupted, 242
 - transportResumed, 242
- activemq::core::ActiveMQConnectionFactory, 244
 - ~ActiveMQConnectionFactory, 245
 - ActiveMQConnectionFactory, 245
 - createConnection, 245, 246
 - getBrokerURL, 247
 - getPassword, 247
 - getUsername, 247
 - setBrokerURL, 247
 - setPassword, 247
 - setUsername, 248
- activemq::core::ActiveMQConnectionMetaData, 249
 - ~ActiveMQConnectionMetaData, 250
 - ActiveMQConnectionMetaData, 250
 - getCMSMajorVersion, 250
 - getCMSMinorVersion, 250
 - getCMSProviderName, 250
 - getCMSVersion, 251
 - getCMSXPropertyNames, 251
 - getProviderMajorVersion, 251
 - getProviderMinorVersion, 251
 - getProviderVersion, 252
- activemq::core::ActiveMQConnectionSupport, 253
 - ~ActiveMQConnectionSupport, 254
 - ActiveMQConnectionSupport, 254
 - getClientId, 255
 - getCloseTimeout, 255
 - getNextLocalTransactionId, 255
 - getNextSessionId, 255
 - getNextTempDestinationId, 255
 - getPassword, 255
 - getProducerWindowSize, 256
 - getProperties, 256
 - getSendTimeout, 256
 - getTransport, 256
 - getUsername, 256
 - isAlwaysSyncSend, 257
 - isUseAsyncSend, 257
 - setAlwaysSyncSend, 257
 - setClientId, 257
 - setCloseTimeout, 257
 - setPassword, 257
 - setProducerWindowSize, 258
 - setSendTimeout, 258
 - setUseAsyncSend, 258
 - setUsername, 258
 - shutdownTransport, 258
 - startUpTransport, 259
- activemq::core::ActiveMQConstants, 260
 - ACK_TYPE_CONSUMED, 261
 - ACK_TYPE_DELIVERED, 261
 - ACK_TYPE_INDIVIDUAL, 261
 - ACK_TYPE_POISON, 261
 - ACK_TYPE_REDELIVERED, 261
 - AckType, 261
 - CONNECTION_ALWAYS_SYNC_SEND, 262
 - CONNECTION_CLOSE_TIMEOUT, 262
 - CONNECTION_PRODUCER_WINDOW_SIZE, 262
 - CONNECTION_SEND_TIMEOUT, 262
 - CONNECTION_USE_ASYNC_SEND, 262
 - CONSUMER_DISPATCH_ASYNC, 261
 - CONSUMER_EXCLUSIVE, 261
 - CONSUMER_NOLOCAL, 261
 - CONSUMER_PREFETCH_SIZE, 261
 - CONSUMER_PRIORITY, 261
 - CONSUMER_RETROACTIVE, 261
 - CONSUMER_SELECTOR, 261
 - CUNSUMER_MAX_PENDING_MSG_LIMIT, 261
 - DESTINATION_ADD_OPERATION, 261

- DESTINATION_REMOVE_ - OPERATION, 261
- DestinationActions, 261
- DestinationOption, 261
- NUM_OPTIONS, 261
- NUM_PARAMS, 262
- PARAM_CLIENTID, 262
- PARAM_PASSWORD, 262
- PARAM_USERNAME, 262
- toDestinationOption, 262
- toString, 262
- toURIOption, 262
- TRANSACTION_STATE_BEGIN, 262
- TRANSACTION_STATE_ - COMMITONEPHASE, 262
- TRANSACTION_STATE_ - COMMITTWO PHASE, 262
- TRANSACTION_STATE_END, 262
- TRANSACTION_STATE_FORGET, 262
- TRANSACTION_STATE_PREPARE, 262
- TRANSACTION_STATE_RECOVER, 262
- TRANSACTION_STATE_ROLLBACK, 262
- TransactionState, 261
- URIParam, 262
- activemq::core::ActiveMQConstants::StaticInitializer, 3016
 - ~StaticInitializer, 3016
 - destOptionMap, 3016
 - destOptions, 3016
 - StaticInitializer, 3016
 - uriParams, 3016
 - uriParamsMap, 3016
- activemq::core::ActiveMQConsumer, 263
 - ~ActiveMQConsumer, 265
 - acknowledge, 265
 - ActiveMQConsumer, 265
 - afterMessageIsConsumed, 265
 - beforeMessageIsConsumed, 266
 - clearMessagesInProgress, 266
 - close, 266
 - commit, 266
 - deliverAcks, 266
 - dequeue, 266
 - dispatch, 267
 - doClose, 267
 - getConsumerId, 267
 - getConsumerInfo, 267
 - getLastDeliveredSequenceId, 268
 - getMessageListener, 268
 - getMessageSelector, 268
 - isClosed, 268
 - isSynchronizationRegistered, 268
 - iterate, 269
 - receive, 269
 - receiveNoWait, 269
 - rollback, 270
 - setLastDeliveredSequenceId, 270
 - setMessageListener, 270
 - setSynchronizationRegistered, 270
 - start, 270
 - stop, 270
- activemq::core::ActiveMQProducer, 406
 - ~ActiveMQProducer, 407
 - ActiveMQProducer, 407
 - close, 408
 - getDeliveryMode, 408
 - getDisableMessageID, 408
 - getDisableMessageTimeStamp, 408
 - getPriority, 408
 - getProducerId, 409
 - getProducerInfo, 409
 - getSendTimeout, 409
 - getTimeToLive, 409
 - isClosed, 409
 - onProducerAck, 410
 - send, 410, 411
 - setDeliveryMode, 412
 - setDisableMessageID, 412
 - setDisableMessageTimeStamp, 412
 - setPriority, 412
 - setSendTimeout, 413
 - setTimeToLive, 413
- activemq::core::ActiveMQSession, 443
 - ~ActiveMQSession, 447
 - acknowledge, 447
 - ActiveMQSession, 447
 - ActiveMQSessionExecutor, 459
 - clearMessagesInProgress, 447
 - close, 447
 - commit, 447
 - createBrowser, 447, 448
 - createBytesMessage, 448
 - createConsumer, 449
 - createDurableConsumer, 450
 - createMapMessage, 450
 - createMessage, 450
 - createProducer, 451
 - createQueue, 451
 - createStreamMessage, 451
 - createTemporaryQueue, 452
 - createTemporaryTopic, 452
 - createTextMessage, 452
 - createTopic, 453
 - deliverAcks, 453
 - dispatch, 453

- disposeOf, 453
- doStartTransaction, 454
- fire, 454
- getAcknowledgeMode, 454
- getConnection, 454
- getExceptionListener, 454
- getLastDeliveredSequenceId, 455
- getSessionId, 455
- getSessionInfo, 455
- isAutoAcknowledge, 455
- isClientAcknowledge, 455
- isDupsOkAcknowledge, 455
- isIndividualAcknowledge, 456
- isStarted, 456
- isTransacted, 456
- oneway, 456
- recover, 456
- redispatch, 457
- rollback, 457
- send, 457
- setLastDeliveredSequenceId, 457
- start, 458
- stop, 458
- syncRequest, 458
- unsubscribe, 458
- wakeup, 458
- activemq::core::ActiveMQSessionExecutor, 460
 - ~ActiveMQSessionExecutor, 461
 - ActiveMQSessionExecutor, 461
 - clear, 461
 - clearMessagesInProgress, 461
 - close, 461
 - execute, 461
 - executeFirst, 461
 - getUnconsumedMessages, 462
 - hasUnconsumedMessages, 462
 - isEmpty, 462
 - isRunning, 462
 - iterate, 462
 - start, 462
 - stop, 462
 - wakeup, 463
- activemq::core::ActiveMQTransactionContext, 622
 - ~ActiveMQTransactionContext, 623
 - ActiveMQTransactionContext, 623
 - addSynchronization, 623
 - begin, 623
 - commit, 623
 - getMaximumRedeliveries, 624
 - getRedeliveryDelay, 624
 - getTransactionId, 624
 - isInTransaction, 624
 - removeSynchronization, 624
 - rollback, 625
- activemq::core::DispatchData, 1535
 - DispatchData, 1535
 - getConsumerId, 1535
 - getMessage, 1535
- activemq::core::Dispatcher, 1536
 - ~Dispatcher, 1536
 - dispatch, 1536
- activemq::core::MessageDispatchChannel, 2224
 - ~MessageDispatchChannel, 2225
 - clear, 2225
 - close, 2225
 - dequeue, 2225
 - dequeueNoWait, 2226
 - enqueue, 2226
 - enqueueFirst, 2226
 - isClosed, 2226
 - isEmpty, 2226
 - isRunning, 2227
 - lock, 2227
 - MessageDispatchChannel, 2225
 - notify, 2227
 - notifyAll, 2227
 - peek, 2228
 - removeAll, 2228
 - size, 2228
 - start, 2228
 - stop, 2228
 - tryLock, 2228
 - unlock, 2229
 - wait, 2229, 2230
- activemq::core::Synchronization, 3137
 - ~Synchronization, 3137
 - afterCommit, 3137
 - afterRollback, 3137
 - beforeEnd, 3137
- activemq::exceptions, 74
- activemq::exceptions::ActiveMQException, 305
 - ~ActiveMQException, 306
 - ActiveMQException, 305, 306
 - clone, 306
 - convertToCMSException, 306
- activemq::exceptions::BrokerException, 749
 - ~BrokerException, 749
 - BrokerException, 749
 - clone, 749
- activemq::io, 75
- activemq::io::LoggingInputStream, 2038
 - ~LoggingInputStream, 2038
 - LoggingInputStream, 2038
 - read, 2038, 2039
- activemq::io::LoggingOutputStream, 2040
 - ~LoggingOutputStream, 2040
 - LoggingOutputStream, 2040

- write, 2040, 2041
- activemq::library, 76
- activemq::library::ActiveMQCPP, 272
 - ~ActiveMQCPP, 272
 - ActiveMQCPP, 272
 - initializeLibrary, 272, 273
 - operator=, 273
 - shutdownLibrary, 273
- activemq::state, 77
- activemq::state::CommandVisitor, 1069
 - ~CommandVisitor, 1071
 - processBeginTransaction, 1071
 - processBrokerError, 1071
 - processBrokerInfo, 1071
 - processCommitTransactionOnePhase, 1071
 - processCommitTransactionTwoPhase, 1071
 - processConnectionControl, 1071
 - processConnectionError, 1072
 - processConnectionInfo, 1072
 - processConsumerControl, 1072
 - processConsumerInfo, 1072
 - processControlCommand, 1072
 - processDestinationInfo, 1072
 - processEndTransaction, 1072
 - processFlushCommand, 1072
 - processForgetTransaction, 1073
 - processKeepAliveInfo, 1073
 - processMessage, 1073
 - processMessageAck, 1073
 - processMessageDispatch, 1073
 - processMessageDispatchNotification, 1073
 - processMessagePull, 1073
 - processPrepareTransaction, 1073
 - processProducerAck, 1073
 - processProducerInfo, 1074
 - processRecoverTransactions, 1074
 - processRemoveConnection, 1074
 - processRemoveConsumer, 1074
 - processRemoveDestination, 1074
 - processRemoveInfo, 1074
 - processRemoveProducer, 1074
 - processRemoveSession, 1074
 - processRemoveSubscriptionInfo, 1075
 - processReplayCommand, 1075
 - processResponse, 1075
 - processRollbackTransaction, 1075
 - processSessionInfo, 1075
 - processShutdownInfo, 1075
 - processTransactionInfo, 1075
 - processWireFormat, 1075
- activemq::state::CommandVisitorAdapter, 1077
 - ~CommandVisitorAdapter, 1080
 - processBeginTransaction, 1080
 - processBrokerError, 1080
 - processBrokerInfo, 1080
 - processCommitTransactionOnePhase, 1080
 - processCommitTransactionTwoPhase, 1080
 - processConnectionControl, 1080
 - processConnectionError, 1080
 - processConnectionInfo, 1080
 - processConsumerControl, 1080
 - processConsumerInfo, 1080
 - processControlCommand, 1080
 - processDestinationInfo, 1080
 - processEndTransaction, 1080
 - processFlushCommand, 1080
 - processForgetTransaction, 1080
 - processKeepAliveInfo, 1080
 - processMessage, 1080
 - processMessageAck, 1080
 - processMessageDispatch, 1080
 - processMessageDispatchNotification, 1080
 - processMessagePull, 1080
 - processPrepareTransaction, 1080
 - processProducerAck, 1080
 - processProducerInfo, 1080
 - processRecoverTransactions, 1080
 - processRemoveConnection, 1080
 - processRemoveConsumer, 1080
 - processRemoveDestination, 1080
 - processRemoveInfo, 1080
 - processRemoveProducer, 1081
 - processRemoveSession, 1081
 - processRemoveSubscriptionInfo, 1081
 - processReplayCommand, 1081
 - processResponse, 1081
 - processRollbackTransaction, 1081
 - processSessionInfo, 1081
 - processShutdownInfo, 1081
 - processTransactionInfo, 1081
 - processWireFormat, 1082
- activemq::state::ConnectionState, 1241
 - ~ConnectionState, 1242
 - addSession, 1242
 - addTempDestination, 1242
 - addTransactionState, 1242
 - checkShutdown, 1242
 - ConnectionState, 1242
 - getInfo, 1242
 - getSessionState, 1242
 - getSessionStates, 1242
 - getTempDestinations, 1242
 - getTransactionState, 1242
 - getTransactionStates, 1242
 - removeSession, 1242
 - removeTempDestination, 1242

- removeTransactionState, 1242
- reset, 1243
- shutdown, 1243
- toString, 1243
- activemq::state::ConnectionStateTracker, 1244
 - ~ConnectionStateTracker, 1246
 - ConnectionStateTracker, 1246
 - getMaxCacheSize, 1246
 - isRestoreConsumers, 1246
 - isRestoreProducers, 1246
 - isRestoreSessions, 1246
 - isRestoreTransaction, 1246
 - isTrackMessages, 1246
 - isTrackTransactions, 1246
 - processBeginTransaction, 1246
 - processCommitTransactionOnePhase, 1246
 - processCommitTransactionTwoPhase, 1246
 - processConnectionInfo, 1246
 - processConsumerInfo, 1247
 - processDestinationInfo, 1247
 - processEndTransaction, 1247
 - processMessage, 1247
 - processMessageAck, 1247
 - processPrepareTransaction, 1247
 - processProducerInfo, 1247
 - processRemoveConnection, 1248
 - processRemoveConsumer, 1248
 - processRemoveDestination, 1248
 - processRemoveProducer, 1248
 - processRemoveSession, 1248
 - processRollbackTransaction, 1248
 - processSessionInfo, 1248
 - RemoveTransactionAction, 1249
 - restore, 1249
 - setMaxCacheSize, 1249
 - setRestoreConsumers, 1249
 - setRestoreProducers, 1249
 - setRestoreSessions, 1249
 - setRestoreTransaction, 1249
 - setTrackMessages, 1249
 - setTrackTransactions, 1249
 - track, 1249
 - trackBack, 1249
- activemq::state::ConsumerState, 1329
 - ~ConsumerState, 1329
 - ConsumerState, 1329
 - getInfo, 1329
 - toString, 1329
- activemq::state::ProducerState, 2656
 - ~ProducerState, 2656
 - getInfo, 2656
 - ProducerState, 2656
 - toString, 2656
- activemq::state::SessionState, 2903
 - ~SessionState, 2904
 - addConsumer, 2904
 - addProducer, 2904
 - checkShutdown, 2904
 - getConsumerState, 2904
 - getConsumerStates, 2904
 - getInfo, 2904
 - getProducerState, 2904
 - getProducerStates, 2904
 - removeConsumer, 2904
 - removeProducer, 2904
 - SessionState, 2904
 - shutdown, 2904
 - toString, 2904
- activemq::state::Tracked, 3216
 - ~Tracked, 3216
 - isWaitingForResponse, 3216
 - onResponse, 3216
 - Tracked, 3216
- activemq::state::TransactionState, 3265
 - ~TransactionState, 3266
 - addCommand, 3266
 - checkShutdown, 3266
 - getCommands, 3266
 - getId, 3266
 - getPreparedResult, 3266
 - isPrepared, 3266
 - setPrepared, 3266
 - setPreparedResult, 3266
 - shutdown, 3266
 - toString, 3266
 - TransactionState, 3266
- activemq::threads, 78
- activemq::threads::CompositeTask, 1090
 - ~CompositeTask, 1090
 - isPending, 1090
- activemq::threads::CompositeTaskRunner, 1092
 - ~CompositeTaskRunner, 1093
 - addTask, 1093
 - CompositeTaskRunner, 1093
 - iterate, 1093
 - removeTask, 1093
 - run, 1093
 - shutdown, 1093
 - wakeup, 1094
- activemq::threads::DedicatedTaskRunner, 1473
 - ~DedicatedTaskRunner, 1473
 - DedicatedTaskRunner, 1473
 - run, 1473
 - shutdown, 1473, 1474
 - wakeup, 1474
- activemq::threads::Task, 3148
 - ~Task, 3148

- iterate, 3148
- activemq::threads::TaskRunner, 3150
 - ~TaskRunner, 3150
 - shutdown, 3150
 - wakeup, 3150
- activemq::transport, 79
- activemq::transport::AbstractTransportFactory, 169
 - ~AbstractTransportFactory, 169
 - createWireFormat, 169
- activemq::transport::CompositeTransport, 1095
 - ~CompositeTransport, 1095
 - addURI, 1095
 - removeURI, 1095
- activemq::transport::correlator, 80
- activemq::transport::correlator::FutureResponse, 1704
 - ~FutureResponse, 1704
 - FutureResponse, 1704
 - getResponse, 1704, 1705
 - setResponse, 1705
- activemq::transport::correlator::ResponseCorrelator, 2787
 - ~ResponseCorrelator, 2788
 - close, 2788
 - onCommand, 2788
 - oneway, 2788
 - onTransportException, 2789
 - request, 2789
 - ResponseCorrelator, 2788
 - start, 2790
- activemq::transport::DefaultTransportListener, 1475
 - ~DefaultTransportListener, 1475
 - onCommand, 1475
 - onException, 1475
 - transportInterrupted, 1476
 - transportResumed, 1476
- activemq::transport::failover, 81
- activemq::transport::failover::BackupTransport, 644
 - ~BackupTransport, 644
 - BackupTransport, 644
 - getTransport, 644
 - getUri, 645
 - isClosed, 645
 - onException, 645
 - setClosed, 645
 - setTransport, 645
 - setUri, 645
- activemq::transport::failover::BackupTransportPool, 647
 - ~BackupTransportPool, 648
 - BackupTransport, 649
 - BackupTransportPool, 648
 - getBackup, 648
 - getBackupPoolSize, 648
 - isEnabled, 648
 - isPending, 648
 - iterate, 649
 - setBackupPoolSize, 649
 - setEnabled, 649
- activemq::transport::failover::CloseTransportsTask, 1022
 - ~CloseTransportsTask, 1022
 - add, 1022
 - CloseTransportsTask, 1022
 - isPending, 1022
 - iterate, 1022
- activemq::transport::failover::FailoverTransport, 1612
 - ~FailoverTransport, 1614
 - add, 1614
 - addURI, 1614
 - close, 1615
 - FailoverTransport, 1614
 - FailoverTransportListener, 1623
 - getBackOffMultiplier, 1615
 - getBackupPoolSize, 1616
 - getInitialReconnectDelay, 1616
 - getMaxCacheSize, 1616
 - getMaxReconnectAttempts, 1616
 - getMaxReconnectDelay, 1616
 - getReconnectDelay, 1616
 - getRemoteAddress, 1616
 - getTimeout, 1616
 - getTransportListener, 1616
 - handleTransportFailure, 1616
 - isBackup, 1617
 - isClosed, 1617
 - isConnected, 1617
 - isFaultTolerant, 1617
 - isInitialized, 1617
 - isPending, 1618
 - isRandomize, 1618
 - isTrackMessages, 1618
 - isUseExponentialBackOff, 1618
 - iterate, 1618
 - narrow, 1618
 - oneway, 1619
 - reconnect, 1619
 - removeURI, 1619
 - request, 1619, 1620
 - restoreTransport, 1620
 - setBackOffMultiplier, 1621
 - setBackup, 1621
 - setBackupPoolSize, 1621
 - setInitialized, 1621

- setInitialReconnectDelay, 1621
- setMaxCacheSize, 1622
- setMaxReconnectAttempts, 1622
- setMaxReconnectDelay, 1622
- setRandomize, 1622
- setReconnectDelay, 1622
- setTimeout, 1622
- setTrackMessages, 1622
- setTransportListener, 1622
- setUseExponentialBackOff, 1622
- setWireFormat, 1622
- start, 1622
- stop, 1623
- activemq::transport::failover::FailoverTransportFactory, 1624
 - ~FailoverTransportFactory, 1625
 - create, 1625
 - createComposite, 1625
 - doCreateComposite, 1625
- activemq::transport::failover::FailoverTransportListener, 1627
 - ~FailoverTransportListener, 1628
 - FailoverTransportListener, 1628
 - onCommand, 1628
 - onException, 1628
 - transportInterrupted, 1628
 - transportResumed, 1628
- activemq::transport::failover::URIPool, 3329
 - ~URIPool, 3329
 - addURI, 3330
 - addURIs, 3330
 - getURI, 3330
 - isRandomize, 3330
 - removeURI, 3330
 - setRandomize, 3331
 - URIPool, 3329
- activemq::transport::inactivity, 82
- activemq::transport::inactivity::InactivityMonitor, 1733
 - ~InactivityMonitor, 1734
 - AsyncSignalReadErrorTask, 1736
 - AsyncWriteTask, 1736
 - close, 1734
 - getInitialDelayTime, 1734
 - getReadCheckTime, 1734
 - getWriteCheckTime, 1734
 - InactivityMonitor, 1734
 - isKeepAliveResponseRequired, 1734
 - onCommand, 1734
 - oneway, 1735
 - onException, 1735
 - ReadChecker, 1736
 - setInitialDelayTime, 1735
 - setKeepAliveResponseRequired, 1736
 - setReadCheckTime, 1736
 - setWriteCheckTime, 1736
 - WriteChecker, 1736
- activemq::transport::inactivity::ReadChecker, 2682
 - ~ReadChecker, 2682
 - ReadChecker, 2682
 - run, 2682
- activemq::transport::inactivity::WriteChecker, 3403
 - ~WriteChecker, 3403
 - run, 3403
 - WriteChecker, 3403
- activemq::transport::IOTransport, 1823
 - ~IOTransport, 1825
 - close, 1825
 - getRemoteAddress, 1825
 - getTransportListener, 1825
 - IOTransport, 1824
 - isClosed, 1825
 - isConnected, 1825
 - isFaultTolerant, 1826
 - narrow, 1826
 - oneway, 1826
 - reconnect, 1826
 - request, 1827
 - run, 1827
 - setInputStream, 1828
 - setOutputStream, 1828
 - setTransportListener, 1828
 - setWireFormat, 1828
 - start, 1828
 - stop, 1829
- activemq::transport::logging, 83
- activemq::transport::logging::LoggingTransport, 2042
 - ~LoggingTransport, 2042
 - LoggingTransport, 2042
 - onCommand, 2043
 - oneway, 2043
 - request, 2043
- activemq::transport::mock, 84
- activemq::transport::mock::InternalCommandListener, 1800
 - ~InternalCommandListener, 1800
 - InternalCommandListener, 1800
 - onCommand, 1800
 - run, 1800
 - setResponseBuilder, 1801
 - setTransport, 1801
- activemq::transport::mock::MockTransport, 2371
 - ~MockTransport, 2373
 - close, 2373

- fireCommand, 2373
- fireException, 2374
- getInstance, 2374
- getNumReceivedMessageBeforeFail, 2374
- getNumReceivedMessages, 2374
- getNumSentKeepAlives, 2374
- getNumSentKeepAlivesBeforeFail, 2374
- getNumSentMessageBeforeFail, 2374
- getNumSentMessages, 2374
- getRemoteAddress, 2374
- getTransportListener, 2374
- getWireFormat, 2375
- isClosed, 2375
- isConnected, 2375
- isFailOnClose, 2375
- isFailOnKeepAliveSends, 2376
- isFailOnReceiveMessage, 2376
- isFailOnSendMessage, 2376
- isFailOnStart, 2376
- isFailOnStop, 2376
- isFaultTolerant, 2376
- MockTransport, 2373
- narrow, 2376
- oneway, 2376
- reconnect, 2377
- request, 2377
- setFailOnClose, 2378
- setFailOnKeepAliveSends, 2379
- setFailOnReceiveMessage, 2379
- setFailOnSendMessage, 2379
- setFailOnStart, 2379
- setFailOnStop, 2379
- setNumReceivedMessageBeforeFail, 2379
- setNumReceivedMessages, 2379
- setNumSentKeepAlives, 2379
- setNumSentKeepAlivesBeforeFail, 2379
- setNumSentMessageBeforeFail, 2379
- setNumSentMessages, 2379
- setOutgoingListener, 2379
- setResponseBuilder, 2379
- setTransportListener, 2380
- setWireFormat, 2380
- start, 2380
- stop, 2380
- activemq::transport::mock::MockTransportFactory, 2382
 - ~MockTransportFactory, 2383
 - create, 2383
 - createComposite, 2383
 - doCreateComposite, 2383
- activemq::transport::mock::ResponseBuilder, 2785
 - ~ResponseBuilder, 2785
 - buildIncomingCommands, 2785
 - buildResponse, 2785
- activemq::transport::tcp, 85
- activemq::transport::tcp::TcpTransport, 3160
 - ~TcpTransport, 3161
 - close, 3161
 - isClosed, 3161
 - isConnected, 3161
 - isFaultTolerant, 3161
 - TcpTransport, 3160, 3161
- activemq::transport::tcp::TcpTransportFactory, 3163
 - ~TcpTransportFactory, 3164
 - create, 3164
 - createComposite, 3164
 - doCreateComposite, 3164
- activemq::transport::Transport, 3273
 - ~Transport, 3274
 - getRemoteAddress, 3274
 - getTransportListener, 3274
 - isClosed, 3274
 - isConnected, 3275
 - isFaultTolerant, 3275
 - narrow, 3275
 - oneway, 3276
 - reconnect, 3276
 - request, 3276, 3277
 - setTransportListener, 3277
 - setWireFormat, 3278
 - start, 3278
 - stop, 3278
- activemq::transport::TransportFactory, 3279
 - ~TransportFactory, 3279
 - create, 3279
 - createComposite, 3280
- activemq::transport::TransportFilter, 3281
 - ~TransportFilter, 3283
 - close, 3283
 - fire, 3283
 - getRemoteAddress, 3283
 - getTransportListener, 3284
 - isClosed, 3284
 - isConnected, 3284
 - isFaultTolerant, 3284
 - listener, 3288
 - narrow, 3285
 - next, 3288
 - onCommand, 3285
 - oneway, 3285
 - onException, 3285
 - reconnect, 3286
 - request, 3286
 - setTransportListener, 3287
 - setWireFormat, 3287
 - start, 3287

- stop, 3288
- TransportFilter, 3283
- transportInterrupted, 3288
- transportResumed, 3288
- activemq::transport::TransportListener, 3289
 - ~TransportListener, 3289
 - onCommand, 3289
 - onException, 3290
 - transportInterrupted, 3290
 - transportResumed, 3290
- activemq::transport::TransportRegistry, 3291
 - ~TransportRegistry, 3292
 - findFactory, 3292
 - getInstance, 3292
 - getTransportNames, 3292
 - registerFactory, 3292
 - unregisterFactory, 3293
- activemq::util, 86
- activemq::util::ActiveMQProperties, 414
 - ~ActiveMQProperties, 415
 - clear, 415
 - clone, 415
 - copy, 415
 - getProperties, 415, 416
 - getProperty, 416
 - hasProperty, 416
 - isEmpty, 416
 - remove, 417
 - setProperties, 417
 - setProperty, 417
 - toArray, 417
 - toString, 417
- activemq::util::CMSExceptionSupport, 1034
 - ~CMSExceptionSupport, 1034
 - create, 1034
 - createMessageEOFException, 1034
 - createMessageFormatException, 1034
- activemq::util::CompositeData, 1088
 - ~CompositeData, 1089
 - CompositeData, 1089
 - getComponents, 1089
 - getFragment, 1089
 - getHost, 1089
 - getParameters, 1089
 - getPath, 1089
 - getScheme, 1089
 - setComponents, 1089
 - setFragment, 1089
 - setHost, 1089
 - setParameters, 1089
 - setPath, 1089
 - setScheme, 1089
 - toURI, 1089
- activemq::util::LongSequenceGenerator, 2090
 - ~LongSequenceGenerator, 2090
 - getLastSequenceId, 2090
 - getNextSequenceId, 2090
 - LongSequenceGenerator, 2090
- activemq::util::MemoryUsage, 2141
 - ~MemoryUsage, 2142
 - decreaseUsage, 2142
 - enqueueUsage, 2142
 - getLimit, 2142
 - getUsage, 2142
 - increaseUsage, 2143
 - isFull, 2143
 - MemoryUsage, 2142
 - setLimit, 2143
 - setUsage, 2143
 - waitForSpace, 2143
- activemq::util::PrimitiveList, 2525
 - ~PrimitiveList, 2527
 - getBool, 2528
 - getByte, 2528
 - getByteArray, 2528
 - getChar, 2529
 - getDouble, 2529
 - getFloat, 2530
 - getInt, 2530
 - getLong, 2530
 - getShort, 2531
 - getString, 2531
 - PrimitiveList, 2527
 - setBool, 2531
 - setByte, 2532
 - setByteArray, 2532
 - setChar, 2532
 - setDouble, 2533
 - setFloat, 2533
 - setInt, 2533
 - setLong, 2534
 - setShort, 2534
 - setString, 2534
 - toString, 2535
- activemq::util::PrimitiveMap, 2536
 - ~PrimitiveMap, 2538
 - getBool, 2539
 - getByte, 2539
 - getByteArray, 2539
 - getChar, 2540
 - getDouble, 2540
 - getFloat, 2540
 - getInt, 2541
 - getLong, 2541
 - getShort, 2542
 - getString, 2542
 - PrimitiveMap, 2538
 - setBool, 2542

- setByte, 2543
- setByteArray, 2543
- setChar, 2543
- setDouble, 2543
- setFloat, 2544
- setInt, 2544
- setLong, 2544
- setShort, 2544
- setString, 2544
- toString, 2545
- activemq::util::PrimitiveValueConverter, 2553
 - ~PrimitiveValueConverter, 2553
 - convert, 2553
 - PrimitiveValueConverter, 2553
- activemq::util::PrimitiveValueNode, 2555
 - ~PrimitiveValueNode, 2561
 - BIG_STRING_TYPE, 2559
 - BOOLEAN_TYPE, 2558
 - BYTE_ARRAY_TYPE, 2559
 - BYTE_TYPE, 2559
 - CHAR_TYPE, 2559
 - clear, 2561
 - DOUBLE_TYPE, 2559
 - FLOAT_TYPE, 2559
 - getBool, 2561
 - getByte, 2562
 - getByteArray, 2562
 - getChar, 2562
 - getDouble, 2562
 - getFloat, 2563
 - getInt, 2563
 - getList, 2563
 - getLong, 2563
 - getMap, 2564
 - getShort, 2564
 - getString, 2564
 - getType, 2564
 - getValue, 2565
 - INTEGER_TYPE, 2559
 - LIST_TYPE, 2559
 - LONG_TYPE, 2559
 - MAP_TYPE, 2559
 - NULL_TYPE, 2558
 - operator=, 2565
 - operator==, 2565
 - PrimitiveType, 2558
 - PrimitiveValueNode, 2559–2561
 - setBool, 2565
 - setByte, 2565
 - setByteArray, 2565
 - setChar, 2566
 - setDouble, 2566
 - setFloat, 2566
 - setInt, 2566
 - setList, 2566
 - setLong, 2567
 - setMap, 2567
 - setShort, 2567
 - setString, 2567
 - setValue, 2567
 - SHORT_TYPE, 2559
 - STRING_TYPE, 2559
 - toString, 2567
- activemq::util::PrimitiveValueNode::PrimitiveValue, 2551
 - boolValue, 2552
 - byteArrayValue, 2552
 - byteValue, 2552
 - charValue, 2552
 - doubleValue, 2552
 - floatValue, 2552
 - intValue, 2552
 - listValue, 2552
 - longValue, 2552
 - mapValue, 2552
 - shortValue, 2552
 - stringValue, 2552
- activemq::util::URISupport, 3332
 - createQueryString, 3332
 - parseComposite, 3332
 - parseQuery, 3333
 - parseURL, 3333
- activemq::util::Usage, 3351
 - ~Usage, 3351
 - decreaseUsage, 3351
 - enqueueUsage, 3351
 - increaseUsage, 3352
 - isFull, 3352
 - waitForSpace, 3352
- activemq::wireformat, 87
- activemq::wireformat::MarshalAware, 2118
 - ~MarshalAware, 2118
 - afterMarshal, 2118
 - afterUnmarshal, 2119
 - beforeMarshal, 2119
 - beforeUnmarshal, 2119
 - getMarshaledForm, 2119
 - isMarshalAware, 2119
 - setMarshaledForm, 2120
- activemq::wireformat::openwire, 88
- activemq::wireformat::openwire::marshal, 89
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshal, 692
 - ~BaseDataStreamMarshaller, 698
 - looseMarshal, 698
 - looseMarshalBrokerError, 698
 - looseMarshalCachedObject, 699
 - looseMarshalLong, 699

- looseMarshalNestedObject, 699
- looseMarshalObjectArray, 700
- looseMarshalString, 700
- looseUnmarshal, 700
- looseUnmarshalBrokerError, 701
- looseUnmarshalByteArray, 701
- looseUnmarshalCachedObject, 701
- looseUnmarshalConstByteArray, 702
- looseUnmarshalLong, 702
- looseUnmarshalNestedObject, 703
- looseUnmarshalString, 703
- readAsciiString, 703
- tightMarshal1, 704
- tightMarshal2, 704
- tightMarshalBrokerError1, 704
- tightMarshalBrokerError2, 705
- tightMarshalCachedObject1, 705
- tightMarshalCachedObject2, 706
- tightMarshalLong1, 706
- tightMarshalLong2, 706
- tightMarshalNestedObject1, 707
- tightMarshalNestedObject2, 707
- tightMarshalObjectArray1, 708
- tightMarshalObjectArray2, 708
- tightMarshalString1, 709
- tightMarshalString2, 709
- tightUnmarshal, 709
- tightUnmarshalBrokerError, 710
- tightUnmarshalByteArray, 710
- tightUnmarshalCachedObject, 710
- tightUnmarshalConstByteArray, 711
- tightUnmarshalLong, 711
- tightUnmarshalNestedObject, 712
- tightUnmarshalString, 712
- toHexFromBytes, 712
- toString, 713
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1418
 - ~DataStreamMarshaller, 1419
 - createObject, 1419
 - getDataStructureType, 1424
 - looseMarshal, 1429
 - looseUnmarshal, 1435
 - tightMarshal1, 1441
 - tightMarshal2, 1447
 - tightUnmarshal, 1453
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2546
 - ~PrimitiveTypesMarshaller, 2547
 - marshal, 2547
 - marshalPrimitive, 2548
 - marshalPrimitiveList, 2548
 - marshalPrimitiveMap, 2548
 - PrimitiveTypesMarshaller, 2547
- unmarshal, 2549
- unmarshalPrimitive, 2549
- unmarshalPrimitiveList, 2550
- unmarshalPrimitiveMap, 2550
- activemq::wireformat::openwire::marshal::v1, 90
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 181
 - ~ActiveMQBlobMessageMarshaller, 182
 - ActiveMQBlobMessageMarshaller, 182
 - createObject, 182
 - getDataStructureType, 182
 - looseMarshal, 182
 - looseUnmarshal, 182
 - tightMarshal1, 183
 - tightMarshal2, 183
 - tightUnmarshal, 184
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 217
 - ~ActiveMQBytesMessageMarshaller, 218
 - ActiveMQBytesMessageMarshaller, 218
 - createObject, 218
 - getDataStructureType, 218
 - looseMarshal, 218
 - looseUnmarshal, 218
 - tightMarshal1, 219
 - tightMarshal2, 219
 - tightUnmarshal, 220
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 289
 - ~ActiveMQDestinationMarshaller, 290
 - ActiveMQDestinationMarshaller, 290
 - looseMarshal, 290
 - looseUnmarshal, 290
 - tightMarshal1, 291
 - tightMarshal2, 291
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 326
 - ~ActiveMQMapMessageMarshaller, 327
 - ActiveMQMapMessageMarshaller, 327
 - createObject, 327
 - getDataStructureType, 327
 - looseMarshal, 327
 - looseUnmarshal, 327
 - tightMarshal1, 328
 - tightMarshal2, 328
 - tightUnmarshal, 329
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 349
 - ~ActiveMQMessageMarshaller, 350
 - ActiveMQMessageMarshaller, 350
 - createObject, 350
 - getDataStructureType, 350

- looseMarshal, 350
- looseUnmarshal, 350
- tightMarshal1, 351
- tightMarshal2, 351
- tightUnmarshal, 352
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 390
 - ~ActiveMQObjectMessageMarshaller, 391
 - ActiveMQObjectMessageMarshaller, 391
 - createObject, 391
 - getDataStructureType, 391
 - looseMarshal, 391
 - looseUnmarshal, 391
 - tightMarshal1, 392
 - tightMarshal2, 392
 - tightUnmarshal, 393
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller, 427
 - ~ActiveMQQueueMarshaller, 428
 - ActiveMQQueueMarshaller, 428
 - createObject, 428
 - getDataStructureType, 428
 - looseMarshal, 428
 - looseUnmarshal, 428
 - tightMarshal1, 429
 - tightMarshal2, 429
 - tightUnmarshal, 430
- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 483
 - ~ActiveMQStreamMessageMarshaller, 484
 - ActiveMQStreamMessageMarshaller, 484
 - createObject, 484
 - getDataStructureType, 484
 - looseMarshal, 484
 - looseUnmarshal, 484
 - tightMarshal1, 485
 - tightMarshal2, 485
 - tightUnmarshal, 486
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 507
 - ~ActiveMQTempDestinationMarshaller, 508
 - ActiveMQTempDestinationMarshaller, 508
 - looseMarshal, 508
 - looseUnmarshal, 508
 - tightMarshal1, 509
 - tightMarshal2, 509
 - tightUnmarshal, 510
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 532
 - ~ActiveMQTempQueueMarshaller, 533
 - ActiveMQTempQueueMarshaller, 533
 - createObject, 533
 - getDataStructureType, 533
- looseMarshal, 533
- looseUnmarshal, 533
- tightMarshal1, 534
- tightMarshal2, 534
- tightUnmarshal, 535
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 557
 - ~ActiveMQTempTopicMarshaller, 558
 - ActiveMQTempTopicMarshaller, 558
 - createObject, 558
 - getDataStructureType, 558
 - looseMarshal, 558
 - looseUnmarshal, 558
 - tightMarshal1, 559
 - tightMarshal2, 559
 - tightUnmarshal, 560
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 582
 - ~ActiveMQTextMessageMarshaller, 583
 - ActiveMQTextMessageMarshaller, 583
 - createObject, 583
 - getDataStructureType, 583
 - looseMarshal, 583
 - looseUnmarshal, 583
 - tightMarshal1, 584
 - tightMarshal2, 584
 - tightUnmarshal, 585
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 606
 - ~ActiveMQTopicMarshaller, 607
 - ActiveMQTopicMarshaller, 607
 - createObject, 607
 - getDataStructureType, 607
 - looseMarshal, 607
 - looseUnmarshal, 607
 - tightMarshal1, 608
 - tightMarshal2, 608
 - tightUnmarshal, 609
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 664
 - ~BaseCommandMarshaller, 665
 - BaseCommandMarshaller, 665
 - looseMarshal, 665
 - looseUnmarshal, 666
 - tightMarshal1, 667
 - tightMarshal2, 668
 - tightUnmarshal, 669
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 760
 - ~BrokerIdMarshaller, 760
 - BrokerIdMarshaller, 760
 - createObject, 760
 - getDataStructureType, 760
 - looseMarshal, 760

- looseUnmarshal, 760
- tightMarshal1, 761
- tightMarshal2, 761
- tightUnmarshal, 762
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 787
 - ~BrokerInfoMarshaller, 788
 - BrokerInfoMarshaller, 788
 - createObject, 788
 - getDataStructureType, 788
 - looseMarshal, 788
 - looseUnmarshal, 788
 - tightMarshal1, 789
 - tightMarshal2, 789
 - tightUnmarshal, 790
- activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1144
 - ~ConnectionControlMarshaller, 1145
 - ConnectionControlMarshaller, 1145
 - createObject, 1145
 - getDataStructureType, 1145
 - looseMarshal, 1145
 - looseUnmarshal, 1145
 - tightMarshal1, 1146
 - tightMarshal2, 1146
 - tightUnmarshal, 1147
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1168
 - ~ConnectionErrorMarshaller, 1169
 - ConnectionErrorMarshaller, 1169
 - createObject, 1169
 - getDataStructureType, 1169
 - looseMarshal, 1169
 - looseUnmarshal, 1169
 - tightMarshal1, 1170
 - tightMarshal2, 1170
 - tightUnmarshal, 1171
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1195
 - ~ConnectionIdMarshaller, 1196
 - ConnectionIdMarshaller, 1196
 - createObject, 1196
 - getDataStructureType, 1196
 - looseMarshal, 1196
 - looseUnmarshal, 1196
 - tightMarshal1, 1197
 - tightMarshal2, 1197
 - tightUnmarshal, 1198
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1225
 - ~ConnectionInfoMarshaller, 1226
 - ConnectionInfoMarshaller, 1226
 - createObject, 1226
 - getDataStructureType, 1226
- looseMarshal, 1226
- looseUnmarshal, 1226
- tightMarshal1, 1227
- tightMarshal2, 1227
- tightUnmarshal, 1228
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1263
 - ~ConsumerControlMarshaller, 1264
 - ConsumerControlMarshaller, 1264
 - createObject, 1264
 - getDataStructureType, 1264
 - looseMarshal, 1264
 - looseUnmarshal, 1264
 - tightMarshal1, 1265
 - tightMarshal2, 1265
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1288
 - ~ConsumerIdMarshaller, 1289
 - ConsumerIdMarshaller, 1289
 - createObject, 1289
 - getDataStructureType, 1289
 - looseMarshal, 1289
 - looseUnmarshal, 1289
 - tightMarshal1, 1290
 - tightMarshal2, 1290
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1313
 - ~ConsumerInfoMarshaller, 1314
 - ConsumerInfoMarshaller, 1314
 - createObject, 1314
 - getDataStructureType, 1314
 - looseMarshal, 1314
 - looseUnmarshal, 1314
 - tightMarshal1, 1315
 - tightMarshal2, 1315
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1342
 - ~ControlCommandMarshaller, 1343
 - ControlCommandMarshaller, 1343
 - createObject, 1343
 - getDataStructureType, 1343
 - looseMarshal, 1343
 - looseUnmarshal, 1343
 - tightMarshal1, 1344
 - tightMarshal2, 1344
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1367
 - ~DataArrayResponseMarshaller, 1368
 - createObject, 1368
 - DataArrayResponseMarshaller, 1368

- getDataStructureType, 1368
 - looseMarshal, 1368
 - looseUnmarshal, 1369
 - tightMarshal1, 1369
 - tightMarshal2, 1369
 - tightUnmarshal, 1370
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1402
 - ~DataResponseMarshaller, 1403
 - createObject, 1403
 - DataResponseMarshaller, 1403
 - getDataStructureType, 1403
 - looseMarshal, 1403
 - looseUnmarshal, 1404
 - tightMarshal1, 1404
 - tightMarshal2, 1404
 - tightUnmarshal, 1405
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1501
 - ~DestinationInfoMarshaller, 1502
 - createObject, 1502
 - DestinationInfoMarshaller, 1502
 - getDataStructureType, 1502
 - looseMarshal, 1502
 - looseUnmarshal, 1502
 - tightMarshal1, 1503
 - tightMarshal2, 1503
 - tightUnmarshal, 1504
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1531
 - ~DiscoveryEventMarshaller, 1532
 - createObject, 1532
 - DiscoveryEventMarshaller, 1532
 - getDataStructureType, 1532
 - looseMarshal, 1532
 - looseUnmarshal, 1532
 - tightMarshal1, 1533
 - tightMarshal2, 1533
 - tightUnmarshal, 1534
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1589
 - ~ExceptionResponseMarshaller, 1590
 - createObject, 1590
 - ExceptionResponseMarshaller, 1590
 - getDataStructureType, 1590
 - looseMarshal, 1590
 - looseUnmarshal, 1591
 - tightMarshal1, 1591
 - tightMarshal2, 1591
 - tightUnmarshal, 1592
- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1683
 - ~FlushCommandMarshaller, 1684
 - createObject, 1684
 - FlushCommandMarshaller, 1684
 - getDataStructureType, 1684
 - looseMarshal, 1684
 - looseUnmarshal, 1684
 - tightMarshal1, 1685
 - tightMarshal2, 1685
 - tightUnmarshal, 1686
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1784
 - ~IntegerResponseMarshaller, 1785
 - createObject, 1785
 - getDataStructureType, 1785
 - IntegerResponseMarshaller, 1785
 - looseMarshal, 1785
 - looseUnmarshal, 1786
 - tightMarshal1, 1786
 - tightMarshal2, 1786
 - tightUnmarshal, 1787
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 1850
 - ~JournalQueueAckMarshaller, 1851
 - createObject, 1851
 - getDataStructureType, 1851
 - JournalQueueAckMarshaller, 1851
 - looseMarshal, 1851
 - looseUnmarshal, 1851
 - tightMarshal1, 1852
 - tightMarshal2, 1852
 - tightUnmarshal, 1853
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 1867
 - ~JournalTopicAckMarshaller, 1868
 - createObject, 1868
 - getDataStructureType, 1868
 - JournalTopicAckMarshaller, 1868
 - looseMarshal, 1868
 - looseUnmarshal, 1868
 - tightMarshal1, 1869
 - tightMarshal2, 1869
 - tightUnmarshal, 1870
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 1898
 - ~JournalTraceMarshaller, 1899
 - createObject, 1899
 - getDataStructureType, 1899
 - JournalTraceMarshaller, 1899
 - looseMarshal, 1899
 - looseUnmarshal, 1899
 - tightMarshal1, 1900
 - tightMarshal2, 1900
 - tightUnmarshal, 1901
- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1922
 - ~JournalTransactionMarshaller, 1923

- createObject, 1923
- getDataStructureType, 1923
- JournalTransactionMarshaller, 1923
- looseMarshal, 1923
- looseUnmarshal, 1923
- tightMarshal1, 1924
- tightMarshal2, 1924
- tightUnmarshal, 1925
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 1949
 - ~KeepAliveInfoMarshaller, 1950
 - createObject, 1950
 - getDataStructureType, 1950
 - KeepAliveInfoMarshaller, 1950
 - looseMarshal, 1950
 - looseUnmarshal, 1950
 - tightMarshal1, 1951
 - tightMarshal2, 1951
 - tightUnmarshal, 1952
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 1965
 - ~LastPartialCommandMarshaller, 1966
 - createObject, 1966
 - getDataStructureType, 1966
 - LastPartialCommandMarshaller, 1966
 - looseMarshal, 1966
 - looseUnmarshal, 1967
 - tightMarshal1, 1967
 - tightMarshal2, 1967
 - tightUnmarshal, 1968
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2009
 - ~LocalTransactionIdMarshaller, 2010
 - createObject, 2010
 - getDataStructureType, 2010
 - LocalTransactionIdMarshaller, 2010
 - looseMarshal, 2010
 - looseUnmarshal, 2010
 - tightMarshal1, 2011
 - tightMarshal2, 2011
 - tightUnmarshal, 2012
- activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2123
 - ~MarshallerFactory, 2123
 - configure, 2123
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2210
 - ~MessageAckMarshaller, 2211
 - createObject, 2211
 - getDataStructureType, 2211
 - looseMarshal, 2211
 - looseUnmarshal, 2211
 - MessageAckMarshaller, 2211
 - tightMarshal1, 2212
 - tightMarshal2, 2212
 - tightUnmarshal, 2213
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2243
 - ~MessageDispatchMarshaller, 2244
 - createObject, 2244
 - getDataStructureType, 2244
 - looseMarshal, 2244
 - looseUnmarshal, 2244
 - MessageDispatchMarshaller, 2244
 - tightMarshal1, 2245
 - tightMarshal2, 2245
 - tightUnmarshal, 2246
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2269
 - ~MessageDispatchNotificationMarshaller, 2270
 - createObject, 2270
 - getDataStructureType, 2270
 - looseMarshal, 2270
 - looseUnmarshal, 2270
 - MessageDispatchNotificationMarshaller, 2270
 - tightMarshal1, 2271
 - tightMarshal2, 2271
 - tightUnmarshal, 2272
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2300
 - ~MessageIdMarshaller, 2301
 - createObject, 2301
 - getDataStructureType, 2301
 - looseMarshal, 2301
 - looseUnmarshal, 2301
 - MessageIdMarshaller, 2301
 - tightMarshal1, 2302
 - tightMarshal2, 2302
 - tightUnmarshal, 2303
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2325
 - ~MessageMarshaller, 2326
 - looseMarshal, 2326
 - looseUnmarshal, 2326
 - MessageMarshaller, 2326
 - tightMarshal1, 2327
 - tightMarshal2, 2327
 - tightUnmarshal, 2328
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2359
 - ~MessagePullMarshaller, 2360
 - createObject, 2360
 - getDataStructureType, 2360
 - looseMarshal, 2360
 - looseUnmarshal, 2360
 - MessagePullMarshaller, 2360

- tightMarshal1, 2361
- tightMarshal2, 2361
- tightUnmarshal, 2362
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2409
 - ~NetworkBridgeFilterMarshaller, 2410
 - createObject, 2410
 - getDataStructureType, 2410
 - looseMarshal, 2410
 - looseUnmarshal, 2410
 - NetworkBridgeFilterMarshaller, 2410
 - tightMarshal1, 2411
 - tightMarshal2, 2411
 - tightUnmarshal, 2412
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2489
 - ~PartialCommandMarshaller, 2490
 - createObject, 2490
 - getDataStructureType, 2490
 - looseMarshal, 2490
 - looseUnmarshal, 2491
 - PartialCommandMarshaller, 2490
 - tightMarshal1, 2491
 - tightMarshal2, 2491
 - tightUnmarshal, 2492
- activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2599
 - ~ProducerAckMarshaller, 2600
 - createObject, 2600
 - getDataStructureType, 2600
 - looseMarshal, 2600
 - looseUnmarshal, 2600
 - ProducerAckMarshaller, 2600
 - tightMarshal1, 2601
 - tightMarshal2, 2601
 - tightUnmarshal, 2602
- activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2623
 - ~ProducerIdMarshaller, 2624
 - createObject, 2624
 - getDataStructureType, 2624
 - looseMarshal, 2624
 - looseUnmarshal, 2624
 - ProducerIdMarshaller, 2624
 - tightMarshal1, 2625
 - tightMarshal2, 2625
 - tightUnmarshal, 2626
- activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2648
 - ~ProducerInfoMarshaller, 2649
 - createObject, 2649
 - getDataStructureType, 2649
 - looseMarshal, 2649
 - looseUnmarshal, 2649
 - ProducerInfoMarshaller, 2649
 - tightMarshal1, 2650
 - tightMarshal2, 2650
 - tightUnmarshal, 2651
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 2707
 - ~RemoveInfoMarshaller, 2708
 - createObject, 2708
 - getDataStructureType, 2708
 - looseMarshal, 2708
 - looseUnmarshal, 2708
 - RemoveInfoMarshaller, 2708
 - tightMarshal1, 2709
 - tightMarshal2, 2709
 - tightUnmarshal, 2710
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 2740
 - ~RemoveSubscriptionInfoMarshaller, 2741
 - createObject, 2741
 - getDataStructureType, 2741
 - looseMarshal, 2741
 - looseUnmarshal, 2741
 - RemoveSubscriptionInfoMarshaller, 2741
 - tightMarshal1, 2742
 - tightMarshal2, 2742
 - tightUnmarshal, 2743
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 2772
 - ~ReplayCommandMarshaller, 2773
 - createObject, 2773
 - getDataStructureType, 2773
 - looseMarshal, 2773
 - looseUnmarshal, 2773
 - ReplayCommandMarshaller, 2773
 - tightMarshal1, 2774
 - tightMarshal2, 2774
 - tightUnmarshal, 2775
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 2806
 - ~ResponseMarshaller, 2807
 - createObject, 2807
 - getDataStructureType, 2807
 - looseMarshal, 2807
 - looseUnmarshal, 2808
 - ResponseMarshaller, 2807
 - tightMarshal1, 2808
 - tightMarshal2, 2809
 - tightUnmarshal, 2809
- activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 2861
 - ~SessionIdMarshaller, 2862
 - createObject, 2862
 - getDataStructureType, 2862
 - looseMarshal, 2862
 - looseUnmarshal, 2862
 - SessionIdMarshaller, 2862
 - tightMarshal1, 2863
 - tightMarshal2, 2863
 - tightUnmarshal, 2864

- looseUnmarshal, 2862
- SessionIdMarshaller, 2862
- tightMarshal1, 2863
- tightMarshal2, 2863
- tightUnmarshal, 2864
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 2881
 - ~SessionInfoMarshaller, 2882
 - createObject, 2882
 - getDataStructureType, 2882
 - looseMarshal, 2882
 - looseUnmarshal, 2882
 - SessionInfoMarshaller, 2882
 - tightMarshal1, 2883
 - tightMarshal2, 2883
 - tightUnmarshal, 2884
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 2937
 - ~ShutdownInfoMarshaller, 2938
 - createObject, 2938
 - getDataStructureType, 2938
 - looseMarshal, 2938
 - looseUnmarshal, 2938
 - ShutdownInfoMarshaller, 2938
 - tightMarshal1, 2939
 - tightMarshal2, 2939
 - tightUnmarshal, 2940
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3102
 - ~SubscriptionInfoMarshaller, 3103
 - createObject, 3103
 - getDataStructureType, 3103
 - looseMarshal, 3103
 - looseUnmarshal, 3103
 - SubscriptionInfoMarshaller, 3103
 - tightMarshal1, 3104
 - tightMarshal2, 3104
 - tightUnmarshal, 3105
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3220
 - ~TransactionIdMarshaller, 3221
 - looseMarshal, 3221
 - looseUnmarshal, 3221
 - tightMarshal1, 3222
 - tightMarshal2, 3222
 - tightUnmarshal, 3223
 - TransactionIdMarshaller, 3221
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3245
 - ~TransactionInfoMarshaller, 3246
 - createObject, 3246
 - getDataStructureType, 3246
 - looseMarshal, 3246
 - looseUnmarshal, 3246
 - tightMarshal1, 3247
 - tightMarshal2, 3247
 - tightUnmarshal, 3248
 - TransactionInfoMarshaller, 3246
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3395
 - ~WireFormatInfoMarshaller, 3396
 - createObject, 3396
 - getDataStructureType, 3396
 - looseMarshal, 3396
 - looseUnmarshal, 3396
 - tightMarshal1, 3397
 - tightMarshal2, 3397
 - tightUnmarshal, 3398
 - WireFormatInfoMarshaller, 3396
- activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3419
 - ~XATransactionIdMarshaller, 3420
 - createObject, 3420
 - getDataStructureType, 3420
 - looseMarshal, 3420
 - looseUnmarshal, 3420
 - tightMarshal1, 3421
 - tightMarshal2, 3421
 - tightUnmarshal, 3422
 - XATransactionIdMarshaller, 3420
- activemq::wireformat::openwire::marshal::v2, 193
 - ~ActiveMQBlobMessageMarshaller, 194
 - ActiveMQBlobMessageMarshaller, 194
 - createObject, 194
 - getDataStructureType, 194
 - looseMarshal, 194
 - looseUnmarshal, 194
 - tightMarshal1, 195
 - tightMarshal2, 195
 - tightUnmarshal, 196
- activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 193
 - ~ActiveMQBlobMessageMarshaller, 194
 - ActiveMQBlobMessageMarshaller, 194
 - createObject, 194
 - getDataStructureType, 194
 - looseMarshal, 194
 - looseUnmarshal, 194
 - tightMarshal1, 195
 - tightMarshal2, 195
 - tightUnmarshal, 196
- activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 229
 - ~ActiveMQBytesMessageMarshaller, 230
 - ActiveMQBytesMessageMarshaller, 230
 - createObject, 230
 - getDataStructureType, 230
 - looseMarshal, 230
 - looseUnmarshal, 230
 - tightMarshal1, 231
 - tightMarshal2, 231
 - tightUnmarshal, 232
- activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 301
 - ~ActiveMQDestinationMarshaller, 302
 - ActiveMQDestinationMarshaller, 302

- looseMarshal, 302
- looseUnmarshal, 302
- tightMarshal1, 303
- tightMarshal2, 303
- tightUnmarshal, 304
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 338
 - ~ActiveMQMapMessageMarshaller, 339
 - ActiveMQMapMessageMarshaller, 339
 - createObject, 339
 - getDataStructureType, 339
 - looseMarshal, 339
 - looseUnmarshal, 339
 - tightMarshal1, 340
 - tightMarshal2, 340
 - tightUnmarshal, 341
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 361
 - ~ActiveMQMessageMarshaller, 362
 - ActiveMQMessageMarshaller, 362
 - createObject, 362
 - getDataStructureType, 362
 - looseMarshal, 362
 - looseUnmarshal, 362
 - tightMarshal1, 363
 - tightMarshal2, 363
 - tightUnmarshal, 364
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 402
 - ~ActiveMQObjectMessageMarshaller, 403
 - ActiveMQObjectMessageMarshaller, 403
 - createObject, 403
 - getDataStructureType, 403
 - looseMarshal, 403
 - looseUnmarshal, 403
 - tightMarshal1, 404
 - tightMarshal2, 404
 - tightUnmarshal, 405
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTextMessageMarshaller, 439
 - ~ActiveMQQueueMarshaller, 440
 - ActiveMQQueueMarshaller, 440
 - createObject, 440
 - getDataStructureType, 440
 - looseMarshal, 440
 - looseUnmarshal, 440
 - tightMarshal1, 441
 - tightMarshal2, 441
 - tightUnmarshal, 442
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 495
 - ~ActiveMQStreamMessageMarshaller, 496
 - ActiveMQStreamMessageMarshaller, 496
 - createObject, 496
 - getDataStructureType, 496
 - looseMarshal, 496
 - looseUnmarshal, 496
 - tightMarshal1, 497
 - tightMarshal2, 497
 - tightUnmarshal, 498
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 519
 - ~ActiveMQTempDestinationMarshaller, 520
 - ActiveMQTempDestinationMarshaller, 520
 - looseMarshal, 520
 - looseUnmarshal, 520
 - tightMarshal1, 521
 - tightMarshal2, 521
 - tightUnmarshal, 522
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 544
 - ~ActiveMQTempQueueMarshaller, 545
 - ActiveMQTempQueueMarshaller, 545
 - createObject, 545
 - getDataStructureType, 545
 - looseMarshal, 545
 - looseUnmarshal, 545
 - tightMarshal1, 546
 - tightMarshal2, 546
 - tightUnmarshal, 547
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 569
 - ~ActiveMQTempTopicMarshaller, 570
 - ActiveMQTempTopicMarshaller, 570
 - createObject, 570
 - getDataStructureType, 570
 - looseMarshal, 570
 - looseUnmarshal, 570
 - tightMarshal1, 571
 - tightMarshal2, 571
 - tightUnmarshal, 572
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTextMessageMarshaller, 594
 - ~ActiveMQTextMessageMarshaller, 595
 - ActiveMQTextMessageMarshaller, 595
 - createObject, 595
 - getDataStructureType, 595
 - looseMarshal, 595
 - looseUnmarshal, 595
 - tightMarshal1, 596
 - tightMarshal2, 596
 - tightUnmarshal, 597
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 618
 - ~ActiveMQTopicMarshaller, 619
 - ActiveMQTopicMarshaller, 619
 - createObject, 619

- getDataStructureType, 619
 - looseMarshal, 619
 - looseUnmarshal, 619
 - tightMarshal1, 620
 - tightMarshal2, 620
 - tightUnmarshal, 621
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 685
 - ~BaseCommandMarshaller, 686
 - BaseCommandMarshaller, 686
 - looseMarshal, 686
 - looseUnmarshal, 687
 - tightMarshal1, 688
 - tightMarshal2, 689
 - tightUnmarshal, 690
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 771
 - ~BrokerIdMarshaller, 772
 - BrokerIdMarshaller, 772
 - createObject, 772
 - getDataStructureType, 772
 - looseMarshal, 772
 - looseUnmarshal, 772
 - tightMarshal1, 773
 - tightMarshal2, 773
 - tightUnmarshal, 774
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 799
 - ~BrokerInfoMarshaller, 800
 - BrokerInfoMarshaller, 800
 - createObject, 800
 - getDataStructureType, 800
 - looseMarshal, 800
 - looseUnmarshal, 800
 - tightMarshal1, 801
 - tightMarshal2, 801
 - tightUnmarshal, 802
- activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1156
 - ~ConnectionControlMarshaller, 1157
 - ConnectionControlMarshaller, 1157
 - createObject, 1157
 - getDataStructureType, 1157
 - looseMarshal, 1157
 - looseUnmarshal, 1157
 - tightMarshal1, 1158
 - tightMarshal2, 1158
 - tightUnmarshal, 1159
- activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1180
 - ~ConnectionErrorMarshaller, 1181
 - ConnectionErrorMarshaller, 1181
 - createObject, 1181
 - getDataStructureType, 1181
 - looseMarshal, 1181
 - looseUnmarshal, 1181
 - tightMarshal1, 1182
 - tightMarshal2, 1182
 - tightUnmarshal, 1183
- activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1207
 - ~ConnectionIdMarshaller, 1208
 - ConnectionIdMarshaller, 1208
 - createObject, 1208
 - getDataStructureType, 1208
 - looseMarshal, 1208
 - looseUnmarshal, 1208
 - tightMarshal1, 1209
 - tightMarshal2, 1209
- activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1233
 - ~ConnectionInfoMarshaller, 1234
 - ConnectionInfoMarshaller, 1234
 - createObject, 1234
 - getDataStructureType, 1234
 - looseMarshal, 1234
 - looseUnmarshal, 1234
 - tightMarshal1, 1235
 - tightMarshal2, 1235
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1271
 - ~ConsumerControlMarshaller, 1272
 - ConsumerControlMarshaller, 1272
 - createObject, 1272
 - getDataStructureType, 1272
 - looseMarshal, 1272
 - looseUnmarshal, 1272
 - tightMarshal1, 1273
 - tightMarshal2, 1273
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1296
 - ~ConsumerIdMarshaller, 1297
 - ConsumerIdMarshaller, 1297
 - createObject, 1297
 - getDataStructureType, 1297
 - looseMarshal, 1297
 - looseUnmarshal, 1297
 - tightMarshal1, 1298
 - tightMarshal2, 1298
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1325
 - ~ConsumerInfoMarshaller, 1326
 - ConsumerInfoMarshaller, 1326
 - createObject, 1326

- getDataStructureType, 1326
 - looseMarshal, 1326
 - looseUnmarshal, 1326
 - tightMarshal1, 1327
 - tightMarshal2, 1327
 - tightUnmarshal, 1328
- activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1350
 - ~ControlCommandMarshaller, 1351
 - ControlCommandMarshaller, 1351
 - createObject, 1351
 - getDataStructureType, 1351
 - looseMarshal, 1351
 - looseUnmarshal, 1351
 - tightMarshal1, 1352
 - tightMarshal2, 1352
 - tightUnmarshal, 1353
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1375
 - ~DataArrayResponseMarshaller, 1376
 - createObject, 1376
 - DataArrayResponseMarshaller, 1376
 - getDataStructureType, 1376
 - looseMarshal, 1376
 - looseUnmarshal, 1377
 - tightMarshal1, 1377
 - tightMarshal2, 1377
 - tightUnmarshal, 1378
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1414
 - ~DataResponseMarshaller, 1415
 - createObject, 1415
 - DataResponseMarshaller, 1415
 - getDataStructureType, 1415
 - looseMarshal, 1415
 - looseUnmarshal, 1416
 - tightMarshal1, 1416
 - tightMarshal2, 1416
 - tightUnmarshal, 1417
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1489
 - ~DestinationInfoMarshaller, 1490
 - createObject, 1490
 - DestinationInfoMarshaller, 1490
 - getDataStructureType, 1490
 - looseMarshal, 1490
 - looseUnmarshal, 1490
 - tightMarshal1, 1491
 - tightMarshal2, 1491
 - tightUnmarshal, 1492
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1515
 - ~DiscoveryEventMarshaller, 1516
 - createObject, 1516
 - DiscoveryEventMarshaller, 1516
 - getDataStructureType, 1516
 - looseMarshal, 1516
 - looseUnmarshal, 1516
 - tightMarshal1, 1517
 - tightMarshal2, 1517
 - tightUnmarshal, 1518
- activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1585
 - ~ExceptionResponseMarshaller, 1586
 - createObject, 1586
 - ExceptionResponseMarshaller, 1586
 - getDataStructureType, 1586
 - looseMarshal, 1586
 - looseUnmarshal, 1587
 - tightMarshal1, 1587
 - tightMarshal2, 1587
 - tightUnmarshal, 1588
- activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1679
 - ~FlushCommandMarshaller, 1680
 - createObject, 1680
 - FlushCommandMarshaller, 1680
 - getDataStructureType, 1680
 - looseMarshal, 1680
 - looseUnmarshal, 1680
 - tightMarshal1, 1681
 - tightMarshal2, 1681
 - tightUnmarshal, 1682
- activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1780
 - ~IntegerResponseMarshaller, 1781
 - createObject, 1781
 - getDataStructureType, 1781
 - IntegerResponseMarshaller, 1781
 - looseMarshal, 1781
 - looseUnmarshal, 1782
 - tightMarshal1, 1782
 - tightMarshal2, 1782
 - tightUnmarshal, 1783
- activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 1838
 - ~JournalQueueAckMarshaller, 1839
 - createObject, 1839
 - getDataStructureType, 1839
 - JournalQueueAckMarshaller, 1839
 - looseMarshal, 1839
 - looseUnmarshal, 1839
 - tightMarshal1, 1840
 - tightMarshal2, 1840
 - tightUnmarshal, 1841
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 1863
 - ~JournalTopicAckMarshaller, 1864

- createObject, 1864
- getDataStructureType, 1864
- JournalTopicAckMarshaller, 1864
- looseMarshal, 1864
- looseUnmarshal, 1864
- tightMarshal1, 1865
- tightMarshal2, 1865
- tightUnmarshal, 1866
- activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 1886
 - ~JournalTraceMarshaller, 1887
 - createObject, 1887
 - getDataStructureType, 1887
 - JournalTraceMarshaller, 1887
 - looseMarshal, 1887
 - looseUnmarshal, 1887
 - tightMarshal1, 1888
 - tightMarshal2, 1888
 - tightUnmarshal, 1889
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1910
 - ~JournalTransactionMarshaller, 1911
 - createObject, 1911
 - getDataStructureType, 1911
 - JournalTransactionMarshaller, 1911
 - looseMarshal, 1911
 - looseUnmarshal, 1911
 - tightMarshal1, 1912
 - tightMarshal2, 1912
 - tightUnmarshal, 1913
- activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 1933
 - ~KeepAliveInfoMarshaller, 1934
 - createObject, 1934
 - getDataStructureType, 1934
 - KeepAliveInfoMarshaller, 1934
 - looseMarshal, 1934
 - looseUnmarshal, 1934
 - tightMarshal1, 1935
 - tightMarshal2, 1935
 - tightUnmarshal, 1936
- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 1961
 - ~LastPartialCommandMarshaller, 1962
 - createObject, 1962
 - getDataStructureType, 1962
 - LastPartialCommandMarshaller, 1962
 - looseMarshal, 1962
 - looseUnmarshal, 1963
 - tightMarshal1, 1963
 - tightMarshal2, 1963
 - tightUnmarshal, 1964
- activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller, 1997
 - ~LocalTransactionMarshaller, 1998
 - createObject, 1998
 - getDataStructureType, 1998
 - LocalTransactionMarshaller, 1998
 - looseMarshal, 1998
 - looseUnmarshal, 1998
 - tightMarshal1, 1999
 - tightMarshal2, 1999
- activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2121
 - ~MarshallerFactory, 2121
 - configure, 2121
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2194
 - ~MessageAckMarshaller, 2195
 - createObject, 2195
 - getDataStructureType, 2195
 - looseMarshal, 2195
 - looseUnmarshal, 2195
 - tightMarshal1, 2196
 - tightMarshal2, 2196
 - tightUnmarshal, 2197
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2231
 - ~MessageDispatchMarshaller, 2232
 - createObject, 2232
 - getDataStructureType, 2232
 - looseMarshal, 2232
 - looseUnmarshal, 2232
 - MessageDispatchMarshaller, 2232
 - tightMarshal1, 2233
 - tightMarshal2, 2233
 - tightUnmarshal, 2234
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2257
 - ~MessageDispatchNotificationMarshaller, 2258
 - createObject, 2258
 - getDataStructureType, 2258
 - looseMarshal, 2258
 - looseUnmarshal, 2258
 - MessageDispatchNotificationMarshaller, 2258
 - tightMarshal1, 2259
 - tightMarshal2, 2259
 - tightUnmarshal, 2260
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2284
 - ~MessageIdMarshaller, 2285
 - createObject, 2285
 - getDataStructureType, 2285
 - looseMarshal, 2285

- looseUnmarshal, 2285
 - MessageIdMarshaller, 2285
 - tightMarshal1, 2286
 - tightMarshal2, 2286
 - tightUnmarshal, 2287
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2305
 - ~MessageMarshaller, 2306
 - looseMarshal, 2306
 - looseUnmarshal, 2306
 - MessageMarshaller, 2306
 - tightMarshal1, 2307
 - tightMarshal2, 2307
 - tightUnmarshal, 2308
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2351
 - ~MessagePullMarshaller, 2352
 - createObject, 2352
 - getDataStructureType, 2352
 - looseMarshal, 2352
 - looseUnmarshal, 2352
 - MessagePullMarshaller, 2352
 - tightMarshal1, 2353
 - tightMarshal2, 2353
 - tightUnmarshal, 2354
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2397
 - ~NetworkBridgeFilterMarshaller, 2398
 - createObject, 2398
 - getDataStructureType, 2398
 - looseMarshal, 2398
 - looseUnmarshal, 2398
 - NetworkBridgeFilterMarshaller, 2398
 - tightMarshal1, 2399
 - tightMarshal2, 2399
 - tightUnmarshal, 2400
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2477
 - ~PartialCommandMarshaller, 2478
 - createObject, 2478
 - getDataStructureType, 2478
 - looseMarshal, 2478
 - looseUnmarshal, 2479
 - PartialCommandMarshaller, 2478
 - tightMarshal1, 2479
 - tightMarshal2, 2479
 - tightUnmarshal, 2480
- activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2583
 - ~ProducerAckMarshaller, 2584
 - createObject, 2584
 - getDataStructureType, 2584
 - looseMarshal, 2584
 - looseUnmarshal, 2584
 - ProducerAckMarshaller, 2584
 - tightMarshal1, 2585
 - tightMarshal2, 2585
 - tightUnmarshal, 2586
- activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2611
 - ~ProducerIdMarshaller, 2612
 - createObject, 2612
 - getDataStructureType, 2612
 - looseMarshal, 2612
 - looseUnmarshal, 2612
 - ProducerIdMarshaller, 2612
 - tightMarshal1, 2613
 - tightMarshal2, 2613
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2636
 - ~ProducerInfoMarshaller, 2637
 - createObject, 2637
 - getDataStructureType, 2637
 - looseMarshal, 2637
 - looseUnmarshal, 2637
 - ProducerInfoMarshaller, 2637
 - tightMarshal1, 2638
 - tightMarshal2, 2638
- activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2711
 - ~RemoveInfoMarshaller, 2712
 - createObject, 2712
 - getDataStructureType, 2712
 - looseMarshal, 2712
 - looseUnmarshal, 2712
 - RemoveInfoMarshaller, 2712
 - tightMarshal1, 2713
 - tightMarshal2, 2713
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 2736
 - ~RemoveSubscriptionInfoMarshaller, 2737
 - createObject, 2737
 - getDataStructureType, 2737
 - looseMarshal, 2737
 - looseUnmarshal, 2737
 - RemoveSubscriptionInfoMarshaller, 2737
 - tightMarshal1, 2738
 - tightMarshal2, 2738
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2756
 - ~ReplayCommandMarshaller, 2757
 - createObject, 2757
 - getDataStructureType, 2757
 - looseMarshal, 2757

- looseUnmarshal, 2757
- ReplayCommandMarshaller, 2757
- tightMarshal1, 2758
- tightMarshal2, 2758
- tightUnmarshal, 2759
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 2791
 - ~ResponseMarshaller, 2792
 - createObject, 2792
 - getDataStructureType, 2792
 - looseMarshal, 2792
 - looseUnmarshal, 2793
 - ResponseMarshaller, 2792
 - tightMarshal1, 2793
 - tightMarshal2, 2794
 - tightUnmarshal, 2794
- activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 2865
 - ~SessionIdMarshaller, 2866
 - createObject, 2866
 - getDataStructureType, 2866
 - looseMarshal, 2866
 - looseUnmarshal, 2866
 - SessionIdMarshaller, 2866
 - tightMarshal1, 2867
 - tightMarshal2, 2867
 - tightUnmarshal, 2868
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2889
 - ~SessionInfoMarshaller, 2890
 - createObject, 2890
 - getDataStructureType, 2890
 - looseMarshal, 2890
 - looseUnmarshal, 2890
 - SessionInfoMarshaller, 2890
 - tightMarshal1, 2891
 - tightMarshal2, 2891
 - tightUnmarshal, 2892
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 2949
 - ~ShutdownInfoMarshaller, 2950
 - createObject, 2950
 - getDataStructureType, 2950
 - looseMarshal, 2950
 - looseUnmarshal, 2950
 - ShutdownInfoMarshaller, 2950
 - tightMarshal1, 2951
 - tightMarshal2, 2951
 - tightUnmarshal, 2952
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3118
 - ~SubscriptionInfoMarshaller, 3119
 - createObject, 3119
 - getDataStructureType, 3119
- looseMarshal, 3119
- looseUnmarshal, 3119
- SubscriptionInfoMarshaller, 3119
- tightMarshal1, 3120
- tightMarshal2, 3120
- TransactionIdMarshaller, 3121
 - ~TransactionIdMarshaller, 3225
 - looseMarshal, 3225
 - looseUnmarshal, 3225
 - tightMarshal1, 3226
 - tightMarshal2, 3226
 - tightUnmarshal, 3227
 - TransactionIdMarshaller, 3225
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3261
 - ~TransactionInfoMarshaller, 3262
 - createObject, 3262
 - getDataStructureType, 3262
 - looseMarshal, 3262
 - looseUnmarshal, 3262
 - tightMarshal1, 3263
 - tightMarshal2, 3263
 - tightUnmarshal, 3264
 - TransactionInfoMarshaller, 3262
- activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3387
 - ~WireFormatInfoMarshaller, 3388
 - createObject, 3388
 - getDataStructureType, 3388
 - looseMarshal, 3388
 - looseUnmarshal, 3388
 - tightMarshal1, 3389
 - tightMarshal2, 3389
 - tightUnmarshal, 3390
 - WireFormatInfoMarshaller, 3388
- activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3427
 - ~XATransactionIdMarshaller, 3428
 - createObject, 3428
 - getDataStructureType, 3428
 - looseMarshal, 3428
 - looseUnmarshal, 3428
 - tightMarshal1, 3429
 - tightMarshal2, 3429
 - tightUnmarshal, 3430
 - XATransactionIdMarshaller, 3428
- activemq::wireformat::openwire::marshal::v3, 177
 - ~ActiveMQBlobMessageMarshaller, 178
 - ActiveMQBlobMessageMarshaller, 178

- createObject, 178
- getDataStructureType, 178
- looseMarshal, 178
- looseUnmarshal, 178
- tightMarshal1, 179
- tightMarshal2, 179
- tightUnmarshal, 180
- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 213
 - ~ActiveMQBytesMessageMarshaller, 214
 - ActiveMQBytesMessageMarshaller, 214
 - createObject, 214
 - getDataStructureType, 214
 - looseMarshal, 214
 - looseUnmarshal, 214
 - tightMarshal1, 215
 - tightMarshal2, 215
 - tightUnmarshal, 216
- activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 285
 - ~ActiveMQDestinationMarshaller, 286
 - ActiveMQDestinationMarshaller, 286
 - looseMarshal, 286
 - looseUnmarshal, 286
 - tightMarshal1, 287
 - tightMarshal2, 287
 - tightUnmarshal, 288
- activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 322
 - ~ActiveMQMapMessageMarshaller, 323
 - ActiveMQMapMessageMarshaller, 323
 - createObject, 323
 - getDataStructureType, 323
 - looseMarshal, 323
 - looseUnmarshal, 323
 - tightMarshal1, 324
 - tightMarshal2, 324
 - tightUnmarshal, 325
- activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 345
 - ~ActiveMQMessageMarshaller, 346
 - ActiveMQMessageMarshaller, 346
 - createObject, 346
 - getDataStructureType, 346
 - looseMarshal, 346
 - looseUnmarshal, 346
 - tightMarshal1, 347
 - tightMarshal2, 347
 - tightUnmarshal, 348
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 386
 - ~ActiveMQObjectMessageMarshaller, 387
 - ActiveMQObjectMessageMarshaller, 387
 - createObject, 387
 - getDataStructureType, 387
 - looseMarshal, 387
 - looseUnmarshal, 387
 - tightMarshal1, 388
 - tightMarshal2, 388
 - tightUnmarshal, 389
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller, 423
 - ~ActiveMQQueueMessageMarshaller, 424
 - ActiveMQQueueMessageMarshaller, 424
 - createObject, 424
 - getDataStructureType, 424
 - looseMarshal, 424
 - looseUnmarshal, 424
 - tightMarshal1, 425
 - tightMarshal2, 425
 - tightUnmarshal, 426
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 479
 - ~ActiveMQStreamMessageMarshaller, 480
 - ActiveMQStreamMessageMarshaller, 480
 - createObject, 480
 - getDataStructureType, 480
 - looseMarshal, 480
 - looseUnmarshal, 480
 - tightMarshal1, 481
 - tightMarshal2, 481
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 503
 - ~ActiveMQTempDestinationMarshaller, 504
 - ActiveMQTempDestinationMarshaller, 504
 - looseMarshal, 504
 - looseUnmarshal, 504
 - tightMarshal1, 505
 - tightMarshal2, 505
 - tightUnmarshal, 506
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 528
 - ~ActiveMQTempQueueMarshaller, 529
 - ActiveMQTempQueueMarshaller, 529
 - createObject, 529
 - getDataStructureType, 529
 - looseMarshal, 529
 - looseUnmarshal, 529
 - tightMarshal1, 530
 - tightMarshal2, 530
 - tightUnmarshal, 531
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 553
 - ~ActiveMQTempTopicMarshaller, 554
 - ActiveMQTempTopicMarshaller, 554
 - createObject, 554

- getDataStructureType, 554
 - looseMarshal, 554
 - looseUnmarshal, 554
 - tightMarshal1, 555
 - tightMarshal2, 555
 - tightUnmarshal, 556
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 578
 - ~ActiveMQTextMessageMarshaller, 579
 - ActiveMQTextMessageMarshaller, 579
 - createObject, 579
 - getDataStructureType, 579
 - looseMarshal, 579
 - looseUnmarshal, 579
 - tightMarshal1, 580
 - tightMarshal2, 580
 - tightUnmarshal, 581
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 602
 - ~ActiveMQTopicMarshaller, 603
 - ActiveMQTopicMarshaller, 603
 - createObject, 603
 - getDataStructureType, 603
 - looseMarshal, 603
 - looseUnmarshal, 603
 - tightMarshal1, 604
 - tightMarshal2, 604
 - tightUnmarshal, 605
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 657
 - ~BaseCommandMarshaller, 658
 - BaseCommandMarshaller, 658
 - looseMarshal, 658
 - looseUnmarshal, 659
 - tightMarshal1, 660
 - tightMarshal2, 661
 - tightUnmarshal, 662
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 755
 - ~BrokerIdMarshaller, 756
 - BrokerIdMarshaller, 756
 - createObject, 756
 - getDataStructureType, 756
 - looseMarshal, 756
 - looseUnmarshal, 756
 - tightMarshal1, 757
 - tightMarshal2, 757
 - tightUnmarshal, 758
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 783
 - ~BrokerInfoMarshaller, 784
 - BrokerInfoMarshaller, 784
 - createObject, 784
 - getDataStructureType, 784
 - looseMarshal, 784
 - looseUnmarshal, 784
 - tightMarshal1, 785
 - tightMarshal2, 785
 - tightUnmarshal, 786
- activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1141
 - ~ConnectionControlMarshaller, 1141
 - ConnectionControlMarshaller, 1141
 - createObject, 1141
 - getDataStructureType, 1141
 - looseMarshal, 1141
 - looseUnmarshal, 1141
 - tightMarshal1, 1142
 - tightMarshal2, 1142
 - tightUnmarshal, 1143
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1165
 - ~ConnectionErrorMarshaller, 1165
 - ConnectionErrorMarshaller, 1165
 - createObject, 1165
 - getDataStructureType, 1165
 - looseMarshal, 1165
 - looseUnmarshal, 1165
 - tightMarshal1, 1166
 - tightMarshal2, 1166
 - tightUnmarshal, 1167
- activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1192
 - ~ConnectionIdMarshaller, 1192
 - ConnectionIdMarshaller, 1192
 - createObject, 1192
 - getDataStructureType, 1192
 - looseMarshal, 1192
 - looseUnmarshal, 1192
 - tightMarshal1, 1193
 - tightMarshal2, 1193
 - tightUnmarshal, 1194
- activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1217
 - ~ConnectionInfoMarshaller, 1218
 - ConnectionInfoMarshaller, 1218
 - createObject, 1218
 - getDataStructureType, 1218
 - looseMarshal, 1218
 - looseUnmarshal, 1218
 - tightMarshal1, 1219
 - tightMarshal2, 1219
 - tightUnmarshal, 1220
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1255
 - ~ConsumerControlMarshaller, 1256
 - ConsumerControlMarshaller, 1256
 - createObject, 1256

- getDataStructureType, 1256
 - looseMarshal, 1256
 - looseUnmarshal, 1256
 - tightMarshal1, 1257
 - tightMarshal2, 1257
 - tightUnmarshal, 1258
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1280
 - ~ConsumerIdMarshaller, 1281
 - ConsumerIdMarshaller, 1281
 - createObject, 1281
 - getDataStructureType, 1281
 - looseMarshal, 1281
 - looseUnmarshal, 1281
 - tightMarshal1, 1282
 - tightMarshal2, 1282
 - tightUnmarshal, 1283
- activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1309
 - ~ConsumerInfoMarshaller, 1310
 - ConsumerInfoMarshaller, 1310
 - createObject, 1310
 - getDataStructureType, 1310
 - looseMarshal, 1310
 - looseUnmarshal, 1310
 - tightMarshal1, 1311
 - tightMarshal2, 1311
 - tightUnmarshal, 1312
- activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1334
 - ~ControlCommandMarshaller, 1335
 - ControlCommandMarshaller, 1335
 - createObject, 1335
 - getDataStructureType, 1335
 - looseMarshal, 1335
 - looseUnmarshal, 1335
 - tightMarshal1, 1336
 - tightMarshal2, 1336
 - tightUnmarshal, 1337
- activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1359
 - ~DataArrayResponseMarshaller, 1360
 - createObject, 1360
 - DataArrayResponseMarshaller, 1360
 - getDataStructureType, 1360
 - looseMarshal, 1360
 - looseUnmarshal, 1361
 - tightMarshal1, 1361
 - tightMarshal2, 1361
 - tightUnmarshal, 1362
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1398
 - ~DataResponseMarshaller, 1399
 - createObject, 1399
 - DataResponseMarshaller, 1399
 - getDataStructureType, 1399
 - looseMarshal, 1399
 - looseUnmarshal, 1400
 - tightMarshal1, 1400
 - tightMarshal2, 1400
 - tightUnmarshal, 1401
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1493
 - ~DestinationInfoMarshaller, 1494
 - createObject, 1494
 - DestinationInfoMarshaller, 1494
 - getDataStructureType, 1494
 - looseMarshal, 1494
 - looseUnmarshal, 1494
 - tightMarshal1, 1495
 - tightMarshal2, 1495
 - tightUnmarshal, 1496
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1519
 - ~DiscoveryEventMarshaller, 1520
 - createObject, 1520
 - DiscoveryEventMarshaller, 1520
 - getDataStructureType, 1520
 - looseMarshal, 1520
 - looseUnmarshal, 1520
 - tightMarshal1, 1521
 - tightMarshal2, 1521
 - tightUnmarshal, 1522
- activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1593
 - ~ExceptionResponseMarshaller, 1594
 - createObject, 1594
 - ExceptionResponseMarshaller, 1594
 - getDataStructureType, 1594
 - looseMarshal, 1594
 - looseUnmarshal, 1595
 - tightMarshal1, 1595
 - tightMarshal2, 1595
 - tightUnmarshal, 1596
- activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1687
 - ~FlushCommandMarshaller, 1688
 - createObject, 1688
 - FlushCommandMarshaller, 1688
 - getDataStructureType, 1688
 - looseMarshal, 1688
 - looseUnmarshal, 1688
 - tightMarshal1, 1689
 - tightMarshal2, 1689
 - tightUnmarshal, 1690
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1788
 - ~IntegerResponseMarshaller, 1789

- createObject, 1789
- getDataStructureType, 1789
- IntegerResponseMarshaller, 1789
- looseMarshal, 1789
- looseUnmarshal, 1790
- tightMarshal1, 1790
- tightMarshal2, 1790
- tightUnmarshal, 1791
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 1846
 - ~JournalQueueAckMarshaller, 1847
 - createObject, 1847
 - getDataStructureType, 1847
 - JournalQueueAckMarshaller, 1847
 - looseMarshal, 1847
 - looseUnmarshal, 1847
 - tightMarshal1, 1848
 - tightMarshal2, 1848
 - tightUnmarshal, 1849
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 1871
 - ~JournalTopicAckMarshaller, 1872
 - createObject, 1872
 - getDataStructureType, 1872
 - JournalTopicAckMarshaller, 1872
 - looseMarshal, 1872
 - looseUnmarshal, 1872
 - tightMarshal1, 1873
 - tightMarshal2, 1873
 - tightUnmarshal, 1874
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 1894
 - ~JournalTraceMarshaller, 1895
 - createObject, 1895
 - getDataStructureType, 1895
 - JournalTraceMarshaller, 1895
 - looseMarshal, 1895
 - looseUnmarshal, 1895
 - tightMarshal1, 1896
 - tightMarshal2, 1896
 - tightUnmarshal, 1897
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1914
 - ~JournalTransactionMarshaller, 1915
 - createObject, 1915
 - getDataStructureType, 1915
 - JournalTransactionMarshaller, 1915
 - looseMarshal, 1915
 - looseUnmarshal, 1915
 - tightMarshal1, 1916
 - tightMarshal2, 1916
 - tightUnmarshal, 1917
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 1941
 - ~KeepAliveInfoMarshaller, 1942
 - createObject, 1942
 - getDataStructureType, 1942
 - KeepAliveInfoMarshaller, 1942
 - looseMarshal, 1942
 - looseUnmarshal, 1942
 - tightMarshal1, 1943
 - tightMarshal2, 1943
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 1969
 - ~LastPartialCommandMarshaller, 1970
 - createObject, 1970
 - getDataStructureType, 1970
 - LastPartialCommandMarshaller, 1970
 - looseMarshal, 1970
 - looseUnmarshal, 1971
 - tightMarshal1, 1971
 - tightMarshal2, 1971
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2001
 - ~LocalTransactionIdMarshaller, 2002
 - createObject, 2002
 - getDataStructureType, 2002
 - LocalTransactionIdMarshaller, 2002
 - looseMarshal, 2002
 - looseUnmarshal, 2002
 - tightMarshal1, 2003
 - tightMarshal2, 2003
- activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2124
 - ~MarshallerFactory, 2124
 - configure, 2124
- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2202
 - ~MessageAckMarshaller, 2203
 - createObject, 2203
 - getDataStructureType, 2203
 - looseMarshal, 2203
 - looseUnmarshal, 2203
 - tightMarshal1, 2204
 - tightMarshal2, 2204
 - tightUnmarshal, 2205
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2239
 - ~MessageDispatchMarshaller, 2240
 - createObject, 2240
 - getDataStructureType, 2240
 - looseMarshal, 2240
 - looseUnmarshal, 2240
 - tightMarshal1, 2240
 - tightMarshal2, 2240
 - tightUnmarshal, 2240

- tightMarshal1, 2241
- tightMarshal2, 2241
- tightUnmarshal, 2242
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2265
 - ~MessageDispatchNotificationMarshaller, 2266
 - createObject, 2266
 - getDataStructureType, 2266
 - looseMarshal, 2266
 - looseUnmarshal, 2266
 - MessageDispatchNotificationMarshaller, 2266
 - tightMarshal1, 2267
 - tightMarshal2, 2267
 - tightUnmarshal, 2268
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2292
 - ~MessageIdMarshaller, 2293
 - createObject, 2293
 - getDataStructureType, 2293
 - looseMarshal, 2293
 - looseUnmarshal, 2293
 - MessageIdMarshaller, 2293
 - tightMarshal1, 2294
 - tightMarshal2, 2294
 - tightUnmarshal, 2295
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2315
 - ~MessageMarshaller, 2316
 - looseMarshal, 2316
 - looseUnmarshal, 2316
 - MessageMarshaller, 2316
 - tightMarshal1, 2317
 - tightMarshal2, 2317
 - tightUnmarshal, 2318
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2355
 - ~MessagePullMarshaller, 2356
 - createObject, 2356
 - getDataStructureType, 2356
 - looseMarshal, 2356
 - looseUnmarshal, 2356
 - MessagePullMarshaller, 2356
 - tightMarshal1, 2357
 - tightMarshal2, 2357
 - tightUnmarshal, 2358
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2401
 - ~NetworkBridgeFilterMarshaller, 2402
 - createObject, 2402
 - getDataStructureType, 2402
 - looseMarshal, 2402
 - looseUnmarshal, 2402
- NetworkBridgeFilterMarshaller, 2402
- tightMarshal1, 2403
- tightMarshal2, 2403
- activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2481
 - ~PartialCommandMarshaller, 2482
 - createObject, 2482
 - getDataStructureType, 2482
 - looseMarshal, 2482
 - looseUnmarshal, 2483
 - PartialCommandMarshaller, 2482
 - tightMarshal1, 2483
 - tightMarshal2, 2483
 - tightUnmarshal, 2484
- activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2587
 - ~ProducerAckMarshaller, 2588
 - createObject, 2588
 - getDataStructureType, 2588
 - looseMarshal, 2588
 - looseUnmarshal, 2588
 - ProducerAckMarshaller, 2588
 - tightMarshal1, 2589
 - tightMarshal2, 2589
 - tightUnmarshal, 2590
- activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2616
 - ~ProducerIdMarshaller, 2616
 - createObject, 2616
 - getDataStructureType, 2616
 - looseMarshal, 2616
 - looseUnmarshal, 2616
 - ProducerIdMarshaller, 2616
 - tightMarshal1, 2617
 - tightMarshal2, 2617
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2652
 - ~ProducerInfoMarshaller, 2653
 - createObject, 2653
 - getDataStructureType, 2653
 - looseMarshal, 2653
 - looseUnmarshal, 2653
 - ProducerInfoMarshaller, 2653
 - tightMarshal1, 2654
 - tightMarshal2, 2654
- activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2715
 - ~RemoveInfoMarshaller, 2716
 - createObject, 2716
 - getDataStructureType, 2716
 - looseMarshal, 2716

- looseUnmarshal, 2716
- RemoveInfoMarshaller, 2716
- tightMarshal1, 2717
- tightMarshal2, 2717
- tightUnmarshal, 2718
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 2744
 - ~RemoveSubscriptionInfoMarshaller, 2745
 - createObject, 2745
 - getDataStructureType, 2745
 - looseMarshal, 2745
 - looseUnmarshal, 2745
 - RemoveSubscriptionInfoMarshaller, 2745
 - tightMarshal1, 2746
 - tightMarshal2, 2746
 - tightUnmarshal, 2747
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2760
 - ~ReplayCommandMarshaller, 2761
 - createObject, 2761
 - getDataStructureType, 2761
 - looseMarshal, 2761
 - looseUnmarshal, 2761
 - ReplayCommandMarshaller, 2761
 - tightMarshal1, 2762
 - tightMarshal2, 2762
 - tightUnmarshal, 2763
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 2796
 - ~ResponseMarshaller, 2797
 - createObject, 2797
 - getDataStructureType, 2797
 - looseMarshal, 2797
 - looseUnmarshal, 2798
 - ResponseMarshaller, 2797
 - tightMarshal1, 2798
 - tightMarshal2, 2799
 - tightUnmarshal, 2799
- activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 2869
 - ~SessionIdMarshaller, 2870
 - createObject, 2870
 - getDataStructureType, 2870
 - looseMarshal, 2870
 - looseUnmarshal, 2870
 - SessionIdMarshaller, 2870
 - tightMarshal1, 2871
 - tightMarshal2, 2871
 - tightUnmarshal, 2872
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2897
 - ~SessionInfoMarshaller, 2898
 - createObject, 2898
 - getDataStructureType, 2898
- looseMarshal, 2898
- looseUnmarshal, 2898
- SessionInfoMarshaller, 2898
- tightMarshal1, 2899
- tightMarshal2, 2899
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 2945
 - ~ShutdownInfoMarshaller, 2946
 - createObject, 2946
 - getDataStructureType, 2946
 - looseMarshal, 2946
 - looseUnmarshal, 2946
 - ShutdownInfoMarshaller, 2946
 - tightMarshal1, 2947
 - tightMarshal2, 2947
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3110
 - ~SubscriptionInfoMarshaller, 3111
 - createObject, 3111
 - getDataStructureType, 3111
 - looseMarshal, 3111
 - looseUnmarshal, 3111
 - SubscriptionInfoMarshaller, 3111
 - tightMarshal1, 3112
 - tightMarshal2, 3112
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3236
 - ~TransactionIdMarshaller, 3237
 - looseMarshal, 3237
 - looseUnmarshal, 3237
 - tightMarshal1, 3238
 - tightMarshal2, 3238
 - tightUnmarshal, 3239
 - TransactionIdMarshaller, 3237
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3253
 - ~TransactionInfoMarshaller, 3254
 - createObject, 3254
 - getDataStructureType, 3254
 - looseMarshal, 3254
 - looseUnmarshal, 3254
 - tightMarshal1, 3255
 - tightMarshal2, 3255
 - tightUnmarshal, 3256
 - TransactionInfoMarshaller, 3254
- activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3379
 - ~WireFormatInfoMarshaller, 3380
 - createObject, 3380
 - getDataStructureType, 3380
 - looseMarshal, 3380

- looseUnmarshal, 3380
 - tightMarshal1, 3381
 - tightMarshal2, 3381
 - tightUnmarshal, 3382
 - WireFormatInfoMarshaller, 3380
- activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3415
 - ~XATransactionIdMarshaller, 3416
 - createObject, 3416
 - getDataStructureType, 3416
 - looseMarshal, 3416
 - looseUnmarshal, 3416
 - tightMarshal1, 3417
 - tightMarshal2, 3417
 - tightUnmarshal, 3418
 - XATransactionIdMarshaller, 3416
- activemq::wireformat::openwire::marshal::v4, 102
- activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 185
 - ~ActiveMQBlobMessageMarshaller, 186
 - ActiveMQBlobMessageMarshaller, 186
 - createObject, 186
 - getDataStructureType, 186
 - looseMarshal, 186
 - looseUnmarshal, 186
 - tightMarshal1, 187
 - tightMarshal2, 187
 - tightUnmarshal, 188
- activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 221
 - ~ActiveMQBytesMessageMarshaller, 222
 - ActiveMQBytesMessageMarshaller, 222
 - createObject, 222
 - getDataStructureType, 222
 - looseMarshal, 222
 - looseUnmarshal, 222
 - tightMarshal1, 223
 - tightMarshal2, 223
 - tightUnmarshal, 224
- activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 293
 - ~ActiveMQDestinationMarshaller, 294
 - ActiveMQDestinationMarshaller, 294
 - looseMarshal, 294
 - looseUnmarshal, 294
 - tightMarshal1, 295
 - tightMarshal2, 295
 - tightUnmarshal, 296
- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 330
 - ~ActiveMQMapMessageMarshaller, 331
 - ActiveMQMapMessageMarshaller, 331
 - createObject, 331
 - getDataStructureType, 331
 - looseMarshal, 331
 - looseUnmarshal, 331
 - tightMarshal1, 332
 - tightMarshal2, 332
 - tightUnmarshal, 333
- activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 353
 - ~ActiveMQMessageMarshaller, 354
 - ActiveMQMessageMarshaller, 354
 - createObject, 354
 - getDataStructureType, 354
 - looseMarshal, 354
 - looseUnmarshal, 354
 - tightMarshal1, 355
 - tightMarshal2, 355
 - tightUnmarshal, 356
- activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 394
 - ~ActiveMQObjectMessageMarshaller, 395
 - ActiveMQObjectMessageMarshaller, 395
 - createObject, 395
 - getDataStructureType, 395
 - looseMarshal, 395
 - looseUnmarshal, 395
 - tightMarshal1, 396
 - tightMarshal2, 396
 - tightUnmarshal, 397
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller, 433
 - ~ActiveMQQueueMarshaller, 432
 - ActiveMQQueueMarshaller, 432
 - createObject, 432
 - getDataStructureType, 432
 - looseMarshal, 432
 - looseUnmarshal, 432
 - tightMarshal1, 433
 - tightMarshal2, 433
 - tightUnmarshal, 434
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 487
 - ~ActiveMQStreamMessageMarshaller, 488
 - ActiveMQStreamMessageMarshaller, 488
 - createObject, 488
 - getDataStructureType, 488
 - looseMarshal, 488
 - looseUnmarshal, 488
 - tightMarshal1, 489
 - tightMarshal2, 489
 - tightUnmarshal, 490
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 511
 - ~ActiveMQTempDestinationMarshaller, 512

- ActiveMQTempDestinationMarshaller, 512
- looseMarshal, 512
- looseUnmarshal, 512
- tightMarshal1, 513
- tightMarshal2, 513
- tightUnmarshal, 514
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 536
 - ~ActiveMQTempDestinationMarshaller, 537
 - ActiveMQTempDestinationMarshaller, 537
 - createObject, 537
 - getDataStructureType, 537
 - looseMarshal, 537
 - looseUnmarshal, 537
 - tightMarshal1, 538
 - tightMarshal2, 538
 - tightUnmarshal, 539
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 561
 - ~ActiveMQTempQueueMarshaller, 562
 - ActiveMQTempQueueMarshaller, 562
 - createObject, 562
 - getDataStructureType, 562
 - looseMarshal, 562
 - looseUnmarshal, 562
 - tightMarshal1, 563
 - tightMarshal2, 563
 - tightUnmarshal, 564
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 586
 - ~ActiveMQTempTopicMarshaller, 587
 - ActiveMQTempTopicMarshaller, 587
 - createObject, 587
 - getDataStructureType, 587
 - looseMarshal, 587
 - looseUnmarshal, 587
 - tightMarshal1, 588
 - tightMarshal2, 588
 - tightUnmarshal, 589
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 610
 - ~ActiveMQTextMessageMarshaller, 611
 - ActiveMQTextMessageMarshaller, 611
 - createObject, 611
 - getDataStructureType, 611
 - looseMarshal, 611
 - looseUnmarshal, 611
 - tightMarshal1, 612
 - tightMarshal2, 612
 - tightUnmarshal, 613
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 671
 - ~BaseCommandMarshaller, 672
 - BaseCommandMarshaller, 672
- looseMarshal, 672
- looseUnmarshal, 673
- tightMarshal1, 674
- tightMarshal2, 675
- tightUnmarshal, 676
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 764
 - ~BrokerIdMarshaller, 764
 - BrokerIdMarshaller, 764
 - createObject, 764
 - getDataStructureType, 764
 - looseMarshal, 764
 - looseUnmarshal, 764
 - tightMarshal1, 765
 - tightMarshal2, 765
 - tightUnmarshal, 766
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 792
 - ~BrokerInfoMarshaller, 792
 - BrokerInfoMarshaller, 792
 - createObject, 792
 - getDataStructureType, 792
 - looseMarshal, 792
 - looseUnmarshal, 792
 - tightMarshal1, 793
 - tightMarshal2, 793
 - tightUnmarshal, 794
- activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1149
 - ~ConnectionControlMarshaller, 1149
 - ConnectionControlMarshaller, 1149
 - createObject, 1149
 - getDataStructureType, 1149
 - looseMarshal, 1149
 - looseUnmarshal, 1149
 - tightMarshal1, 1150
 - tightMarshal2, 1150
 - tightUnmarshal, 1151
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1173
 - ~ConnectionErrorMarshaller, 1173
 - ConnectionErrorMarshaller, 1173
 - createObject, 1173
 - getDataStructureType, 1173
 - looseMarshal, 1173
 - looseUnmarshal, 1173
 - tightMarshal1, 1174
 - tightMarshal2, 1174
 - tightUnmarshal, 1175
- activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1200
 - ~ConnectionIdMarshaller, 1200
 - ConnectionIdMarshaller, 1200
 - createObject, 1200

- getDataStructureType, 1200
 - looseMarshal, 1200
 - looseUnmarshal, 1200
 - tightMarshal1, 1201
 - tightMarshal2, 1201
 - tightUnmarshal, 1202
- activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1221
 - ~ConnectionInfoMarshaller, 1222
 - ConnectionInfoMarshaller, 1222
 - createObject, 1222
 - getDataStructureType, 1222
 - looseMarshal, 1222
 - looseUnmarshal, 1222
 - tightMarshal1, 1223
 - tightMarshal2, 1223
 - tightUnmarshal, 1224
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1259
 - ~ConsumerControlMarshaller, 1260
 - ConsumerControlMarshaller, 1260
 - createObject, 1260
 - getDataStructureType, 1260
 - looseMarshal, 1260
 - looseUnmarshal, 1260
 - tightMarshal1, 1261
 - tightMarshal2, 1261
 - tightUnmarshal, 1262
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1284
 - ~ConsumerIdMarshaller, 1285
 - ConsumerIdMarshaller, 1285
 - createObject, 1285
 - getDataStructureType, 1285
 - looseMarshal, 1285
 - looseUnmarshal, 1285
 - tightMarshal1, 1286
 - tightMarshal2, 1286
 - tightUnmarshal, 1287
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1317
 - ~ConsumerInfoMarshaller, 1318
 - ConsumerInfoMarshaller, 1318
 - createObject, 1318
 - getDataStructureType, 1318
 - looseMarshal, 1318
 - looseUnmarshal, 1318
 - tightMarshal1, 1319
 - tightMarshal2, 1319
 - tightUnmarshal, 1320
- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1338
 - ~ControlCommandMarshaller, 1339
 - ControlCommandMarshaller, 1339
 - createObject, 1339
 - getDataStructureType, 1339
 - looseMarshal, 1339
 - looseUnmarshal, 1339
 - tightMarshal1, 1340
 - tightMarshal2, 1340
 - tightUnmarshal, 1341
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1363
 - ~DataArrayResponseMarshaller, 1364
 - createObject, 1364
 - DataArrayResponseMarshaller, 1364
 - getDataStructureType, 1364
 - looseMarshal, 1364
 - looseUnmarshal, 1365
 - tightMarshal1, 1365
 - tightMarshal2, 1365
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1410
 - ~DataResponseMarshaller, 1411
 - createObject, 1411
 - DataResponseMarshaller, 1411
 - getDataStructureType, 1411
 - looseMarshal, 1411
 - looseUnmarshal, 1412
 - tightMarshal1, 1412
 - tightMarshal2, 1412
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1497
 - ~DestinationInfoMarshaller, 1498
 - createObject, 1498
 - DestinationInfoMarshaller, 1498
 - getDataStructureType, 1498
 - looseMarshal, 1498
 - looseUnmarshal, 1498
 - tightMarshal1, 1499
 - tightMarshal2, 1499
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1523
 - ~DiscoveryEventMarshaller, 1524
 - createObject, 1524
 - DiscoveryEventMarshaller, 1524
 - getDataStructureType, 1524
 - looseMarshal, 1524
 - looseUnmarshal, 1524
 - tightMarshal1, 1525
 - tightMarshal2, 1525
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1597
 - ~ExceptionResponseMarshaller, 1598

- createObject, 1598
- ExceptionResponseMarshaller, 1598
- getDataStructureType, 1598
- looseMarshal, 1598
- looseUnmarshal, 1599
- tightMarshal1, 1599
- tightMarshal2, 1599
- tightUnmarshal, 1600
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1691
 - ~FlushCommandMarshaller, 1692
 - createObject, 1692
 - FlushCommandMarshaller, 1692
 - getDataStructureType, 1692
 - looseMarshal, 1692
 - looseUnmarshal, 1692
 - tightMarshal1, 1693
 - tightMarshal2, 1693
 - tightUnmarshal, 1694
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1792
 - ~IntegerResponseMarshaller, 1793
 - createObject, 1793
 - getDataStructureType, 1793
 - IntegerResponseMarshaller, 1793
 - looseMarshal, 1793
 - looseUnmarshal, 1794
 - tightMarshal1, 1794
 - tightMarshal2, 1794
 - tightUnmarshal, 1795
- activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 1842
 - ~JournalQueueAckMarshaller, 1843
 - createObject, 1843
 - getDataStructureType, 1843
 - JournalQueueAckMarshaller, 1843
 - looseMarshal, 1843
 - looseUnmarshal, 1843
 - tightMarshal1, 1844
 - tightMarshal2, 1844
 - tightUnmarshal, 1845
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 1875
 - ~JournalTopicAckMarshaller, 1876
 - createObject, 1876
 - getDataStructureType, 1876
 - JournalTopicAckMarshaller, 1876
 - looseMarshal, 1876
 - looseUnmarshal, 1876
 - tightMarshal1, 1877
 - tightMarshal2, 1877
 - tightUnmarshal, 1878
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1890
 - ~JournalTransactionMarshaller, 1891
 - createObject, 1891
 - getDataStructureType, 1891
 - JournalTransactionMarshaller, 1891
 - looseMarshal, 1891
 - looseUnmarshal, 1891
 - tightMarshal1, 1892
 - tightMarshal2, 1892
 - tightUnmarshal, 1893
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1918
 - ~JournalTransactionMarshaller, 1919
 - createObject, 1919
 - getDataStructureType, 1919
 - JournalTransactionMarshaller, 1919
 - looseMarshal, 1919
 - looseUnmarshal, 1919
 - tightMarshal1, 1920
 - tightMarshal2, 1920
 - tightUnmarshal, 1921
- activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 1945
 - ~KeepAliveInfoMarshaller, 1946
 - createObject, 1946
 - getDataStructureType, 1946
 - KeepAliveInfoMarshaller, 1946
 - looseMarshal, 1946
 - looseUnmarshal, 1946
 - tightMarshal1, 1947
 - tightMarshal2, 1947
 - tightUnmarshal, 1948
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 1973
 - ~LastPartialCommandMarshaller, 1974
 - createObject, 1974
 - getDataStructureType, 1974
 - LastPartialCommandMarshaller, 1974
 - looseMarshal, 1974
 - looseUnmarshal, 1975
 - tightMarshal1, 1975
 - tightMarshal2, 1975
 - tightUnmarshal, 1976
- activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2005
 - ~LocalTransactionIdMarshaller, 2006
 - createObject, 2006
 - getDataStructureType, 2006
 - LocalTransactionIdMarshaller, 2006
 - looseMarshal, 2006
 - looseUnmarshal, 2006
 - tightMarshal1, 2007
 - tightMarshal2, 2007
 - tightUnmarshal, 2008

- activemq::wireformat::openwire::marshal::v4::MessageAckUnmarshaller, 2122
 - ~MarshallerFactory, 2122
 - configure, 2122
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2206
 - ~MessageAckMarshaller, 2207
 - createObject, 2207
 - getDataStructureType, 2207
 - looseMarshal, 2207
 - looseUnmarshal, 2207
 - MessageAckMarshaller, 2207
 - tightMarshal1, 2208
 - tightMarshal2, 2208
 - tightUnmarshal, 2209
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2235
 - ~MessageDispatchMarshaller, 2236
 - createObject, 2236
 - getDataStructureType, 2236
 - looseMarshal, 2236
 - looseUnmarshal, 2236
 - MessageDispatchMarshaller, 2236
 - tightMarshal1, 2237
 - tightMarshal2, 2237
 - tightUnmarshal, 2238
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2261
 - ~MessageDispatchNotificationMarshaller, 2262
 - createObject, 2262
 - getDataStructureType, 2262
 - looseMarshal, 2262
 - looseUnmarshal, 2262
 - MessageDispatchNotificationMarshaller, 2262
 - tightMarshal1, 2263
 - tightMarshal2, 2263
 - tightUnmarshal, 2264
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2288
 - ~MessageIdMarshaller, 2289
 - createObject, 2289
 - getDataStructureType, 2289
 - looseMarshal, 2289
 - looseUnmarshal, 2289
 - MessageIdMarshaller, 2289
 - tightMarshal1, 2290
 - tightMarshal2, 2290
 - tightUnmarshal, 2291
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2310
 - ~MessageMarshaller, 2311
 - looseMarshal, 2311
 - looseUnmarshal, 2311
 - MessageMarshaller, 2311
 - tightMarshal1, 2312
 - tightMarshal2, 2312
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2367
 - ~MessagePullMarshaller, 2368
 - createObject, 2368
 - getDataStructureType, 2368
 - looseMarshal, 2368
 - looseUnmarshal, 2368
 - MessagePullMarshaller, 2368
 - tightMarshal1, 2369
 - tightMarshal2, 2369
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2413
 - ~NetworkBridgeFilterMarshaller, 2414
 - createObject, 2414
 - getDataStructureType, 2414
 - looseMarshal, 2414
 - looseUnmarshal, 2414
 - NetworkBridgeFilterMarshaller, 2414
 - tightMarshal1, 2415
 - tightMarshal2, 2415
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2485
 - ~PartialCommandMarshaller, 2486
 - createObject, 2486
 - getDataStructureType, 2486
 - looseMarshal, 2486
 - looseUnmarshal, 2487
 - PartialCommandMarshaller, 2486
 - tightMarshal1, 2487
 - tightMarshal2, 2487
 - tightUnmarshal, 2488
- activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2592
 - ~ProducerAckMarshaller, 2592
 - createObject, 2592
 - getDataStructureType, 2592
 - looseMarshal, 2592
 - looseUnmarshal, 2592
 - ProducerAckMarshaller, 2592
 - tightMarshal1, 2593
 - tightMarshal2, 2593
 - tightUnmarshal, 2594
- activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2628
 - ~ProducerIdMarshaller, 2628
 - createObject, 2628
 - getDataStructureType, 2628

- looseMarshal, 2628
- looseUnmarshal, 2628
- ProducerIdMarshaller, 2628
- tightMarshal1, 2629
- tightMarshal2, 2629
- tightUnmarshal, 2630
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2644
 - ~ProducerInfoMarshaller, 2645
 - createObject, 2645
 - getDataSetType, 2645
 - looseMarshal, 2645
 - looseUnmarshal, 2645
 - ProducerInfoMarshaller, 2645
 - tightMarshal1, 2646
 - tightMarshal2, 2646
 - tightUnmarshal, 2647
- activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 2719
 - ~RemoveInfoMarshaller, 2720
 - createObject, 2720
 - getDataSetType, 2720
 - looseMarshal, 2720
 - looseUnmarshal, 2720
 - RemoveInfoMarshaller, 2720
 - tightMarshal1, 2721
 - tightMarshal2, 2721
 - tightUnmarshal, 2722
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 2748
 - ~RemoveSubscriptionInfoMarshaller, 2749
 - createObject, 2749
 - getDataSetType, 2749
 - looseMarshal, 2749
 - looseUnmarshal, 2749
 - RemoveSubscriptionInfoMarshaller, 2749
 - tightMarshal1, 2750
 - tightMarshal2, 2750
 - tightUnmarshal, 2751
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2768
 - ~ReplayCommandMarshaller, 2769
 - createObject, 2769
 - getDataSetType, 2769
 - looseMarshal, 2769
 - looseUnmarshal, 2769
 - ReplayCommandMarshaller, 2769
 - tightMarshal1, 2770
 - tightMarshal2, 2770
 - tightUnmarshal, 2771
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 2811
 - ~ResponseMarshaller, 2812
 - createObject, 2812
 - getDataSetType, 2812
 - looseMarshal, 2812
 - looseUnmarshal, 2813
 - ResponseMarshaller, 2812
 - tightMarshal1, 2813
 - tightMarshal2, 2814
 - tightUnmarshal, 2814
- activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 2873
 - ~SessionIdMarshaller, 2874
 - createObject, 2874
 - getDataSetType, 2874
 - looseMarshal, 2874
 - looseUnmarshal, 2874
 - SessionIdMarshaller, 2874
 - tightMarshal1, 2875
 - tightMarshal2, 2875
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2885
 - ~SessionInfoMarshaller, 2886
 - createObject, 2886
 - getDataSetType, 2886
 - looseMarshal, 2886
 - looseUnmarshal, 2886
 - SessionInfoMarshaller, 2886
 - tightMarshal1, 2887
 - tightMarshal2, 2887
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 2941
 - ~ShutdownInfoMarshaller, 2942
 - createObject, 2942
 - getDataSetType, 2942
 - looseMarshal, 2942
 - looseUnmarshal, 2942
 - ShutdownInfoMarshaller, 2942
 - tightMarshal1, 2943
 - tightMarshal2, 2943
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3106
 - ~SubscriptionInfoMarshaller, 3107
 - createObject, 3107
 - getDataSetType, 3107
 - looseMarshal, 3107
 - looseUnmarshal, 3107
 - SubscriptionInfoMarshaller, 3107
 - tightMarshal1, 3108
 - tightMarshal2, 3108
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3228
 - ~TransactionIdMarshaller, 3229

- looseMarshal, 3229
- looseUnmarshal, 3229
- tightMarshal1, 3230
- tightMarshal2, 3230
- tightUnmarshal, 3231
- TransactionIdMarshaller, 3229
- activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3249
 - ~TransactionInfoMarshaller, 3250
 - createObject, 3250
 - getDataStructureType, 3250
 - looseMarshal, 3250
 - looseUnmarshal, 3250
 - tightMarshal1, 3251
 - tightMarshal2, 3251
 - tightUnmarshal, 3252
 - TransactionInfoMarshaller, 3250
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3391
 - ~WireFormatInfoMarshaller, 3392
 - createObject, 3392
 - getDataStructureType, 3392
 - looseMarshal, 3392
 - looseUnmarshal, 3392
 - tightMarshal1, 3393
 - tightMarshal2, 3393
 - tightUnmarshal, 3394
 - WireFormatInfoMarshaller, 3392
- activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3423
 - ~XATransactionIdMarshaller, 3424
 - createObject, 3424
 - getDataStructureType, 3424
 - looseMarshal, 3424
 - looseUnmarshal, 3424
 - tightMarshal1, 3425
 - tightMarshal2, 3425
 - tightUnmarshal, 3426
 - XATransactionIdMarshaller, 3424
- activemq::wireformat::openwire::marshal::v5, 106
- activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 189
 - ~ActiveMQBlobMessageMarshaller, 190
 - ActiveMQBlobMessageMarshaller, 190
 - createObject, 190
 - getDataStructureType, 190
 - looseMarshal, 190
 - looseUnmarshal, 190
 - tightMarshal1, 191
 - tightMarshal2, 191
 - tightUnmarshal, 192
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 225
 - ~ActiveMQBytesMessageMarshaller, 226
 - ActiveMQBytesMessageMarshaller, 226
 - createObject, 226
 - getDataStructureType, 226
 - looseMarshal, 226
 - looseUnmarshal, 226
 - tightMarshal1, 227
 - tightMarshal2, 227
 - tightUnmarshal, 228
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 297
 - ~ActiveMQDestinationMarshaller, 298
 - ActiveMQDestinationMarshaller, 298
 - looseMarshal, 298
 - looseUnmarshal, 298
 - tightMarshal1, 299
 - tightMarshal2, 299
 - tightUnmarshal, 300
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 334
 - ~ActiveMQMapMessageMarshaller, 335
 - ActiveMQMapMessageMarshaller, 335
 - createObject, 335
 - getDataStructureType, 335
 - looseMarshal, 335
 - looseUnmarshal, 335
 - tightMarshal1, 336
 - tightMarshal2, 336
 - tightUnmarshal, 337
- activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 357
 - ~ActiveMQMessageMarshaller, 358
 - ActiveMQMessageMarshaller, 358
 - createObject, 358
 - getDataStructureType, 358
 - looseMarshal, 358
 - looseUnmarshal, 358
 - tightMarshal1, 359
 - tightMarshal2, 359
 - tightUnmarshal, 360
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 398
 - ~ActiveMQObjectMessageMarshaller, 399
 - ActiveMQObjectMessageMarshaller, 399
 - createObject, 399
 - getDataStructureType, 399
 - looseMarshal, 399
 - looseUnmarshal, 399
 - tightMarshal1, 400
 - tightMarshal2, 400
 - tightUnmarshal, 401
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 435
 - ~ActiveMQQueueMarshaller, 436

- ActiveMQQueueMarshaller, 436
- createObject, 436
- getDataStructureType, 436
- looseMarshal, 436
- looseUnmarshal, 436
- tightMarshal1, 437
- tightMarshal2, 437
- tightUnmarshal, 438
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 491
 - ~ActiveMQStreamMessageMarshaller, 492
 - ActiveMQStreamMessageMarshaller, 492
 - createObject, 492
 - getDataStructureType, 492
 - looseMarshal, 492
 - looseUnmarshal, 492
 - tightMarshal1, 493
 - tightMarshal2, 493
 - tightUnmarshal, 494
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 515
 - ~ActiveMQTempDestinationMarshaller, 516
 - ActiveMQTempDestinationMarshaller, 516
 - looseMarshal, 516
 - looseUnmarshal, 516
 - tightMarshal1, 517
 - tightMarshal2, 517
 - tightUnmarshal, 518
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 540
 - ~ActiveMQTempQueueMarshaller, 541
 - ActiveMQTempQueueMarshaller, 541
 - createObject, 541
 - getDataStructureType, 541
 - looseMarshal, 541
 - looseUnmarshal, 541
 - tightMarshal1, 542
 - tightMarshal2, 542
 - tightUnmarshal, 543
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 565
 - ~ActiveMQTempTopicMarshaller, 566
 - ActiveMQTempTopicMarshaller, 566
 - createObject, 566
 - getDataStructureType, 566
 - looseMarshal, 566
 - looseUnmarshal, 566
 - tightMarshal1, 567
 - tightMarshal2, 567
 - tightUnmarshal, 568
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 590
 - ~ActiveMQTextMessageMarshaller, 591
- ActiveMQTextMessageMarshaller, 591
- createObject, 591
- getDataStructureType, 591
- looseMarshal, 591
- looseUnmarshal, 591
- tightMarshal1, 592
- tightMarshal2, 592
- tightUnmarshal, 593
- activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 614
 - ~ActiveMQTopicMarshaller, 615
 - ActiveMQTopicMarshaller, 615
 - createObject, 615
 - getDataStructureType, 615
 - looseMarshal, 615
 - looseUnmarshal, 615
 - tightMarshal1, 616
 - tightMarshal2, 616
 - tightUnmarshal, 617
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 678
 - ~BaseCommandMarshaller, 679
 - BaseCommandMarshaller, 679
 - looseMarshal, 679
 - looseUnmarshal, 680
 - tightMarshal1, 681
 - tightMarshal2, 682
 - tightUnmarshal, 683
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 767
 - ~BrokerIdMarshaller, 768
 - BrokerIdMarshaller, 768
 - createObject, 768
 - getDataStructureType, 768
 - looseMarshal, 768
 - looseUnmarshal, 768
 - tightMarshal1, 769
 - tightMarshal2, 769
 - tightUnmarshal, 770
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 795
 - ~BrokerInfoMarshaller, 796
 - BrokerInfoMarshaller, 796
 - createObject, 796
 - getDataStructureType, 796
 - looseMarshal, 796
 - looseUnmarshal, 796
 - tightMarshal1, 797
 - tightMarshal2, 797
 - tightUnmarshal, 798
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1152
 - ~ConnectionControlMarshaller, 1153
 - ConnectionControlMarshaller, 1153

- createObject, 1153
- getDataStructureType, 1153
- looseMarshal, 1153
- looseUnmarshal, 1153
- tightMarshal1, 1154
- tightMarshal2, 1154
- tightUnmarshal, 1155
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1176
 - ~ConnectionErrorMarshaller, 1177
 - ConnectionErrorMarshaller, 1177
 - createObject, 1177
 - getDataStructureType, 1177
 - looseMarshal, 1177
 - looseUnmarshal, 1177
 - tightMarshal1, 1178
 - tightMarshal2, 1178
 - tightUnmarshal, 1179
- activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1203
 - ~ConnectionIdMarshaller, 1204
 - ConnectionIdMarshaller, 1204
 - createObject, 1204
 - getDataStructureType, 1204
 - looseMarshal, 1204
 - looseUnmarshal, 1204
 - tightMarshal1, 1205
 - tightMarshal2, 1205
 - tightUnmarshal, 1206
- activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1229
 - ~ConnectionInfoMarshaller, 1230
 - ConnectionInfoMarshaller, 1230
 - createObject, 1230
 - getDataStructureType, 1230
 - looseMarshal, 1230
 - looseUnmarshal, 1230
 - tightMarshal1, 1231
 - tightMarshal2, 1231
 - tightUnmarshal, 1232
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1267
 - ~ConsumerControlMarshaller, 1268
 - ConsumerControlMarshaller, 1268
 - createObject, 1268
 - getDataStructureType, 1268
 - looseMarshal, 1268
 - looseUnmarshal, 1268
 - tightMarshal1, 1269
 - tightMarshal2, 1269
 - tightUnmarshal, 1270
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1292
 - ~ConsumerIdMarshaller, 1293
 - ConsumerIdMarshaller, 1293
 - createObject, 1293
 - getDataStructureType, 1293
 - looseMarshal, 1293
 - looseUnmarshal, 1293
 - tightMarshal1, 1294
 - tightMarshal2, 1294
 - tightUnmarshal, 1295
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1321
 - ~ConsumerInfoMarshaller, 1322
 - ConsumerInfoMarshaller, 1322
 - createObject, 1322
 - getDataStructureType, 1322
 - looseMarshal, 1322
 - looseUnmarshal, 1322
 - tightMarshal1, 1323
 - tightMarshal2, 1323
 - tightUnmarshal, 1324
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1346
 - ~ControlCommandMarshaller, 1347
 - ControlCommandMarshaller, 1347
 - createObject, 1347
 - getDataStructureType, 1347
 - looseMarshal, 1347
 - looseUnmarshal, 1347
 - tightMarshal1, 1348
 - tightMarshal2, 1348
 - tightUnmarshal, 1349
- activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1371
 - ~DataArrayResponseMarshaller, 1372
 - DataArrayResponseMarshaller, 1372
 - createObject, 1372
 - getDataStructureType, 1372
 - looseMarshal, 1372
 - looseUnmarshal, 1373
 - tightMarshal1, 1373
 - tightMarshal2, 1373
 - tightUnmarshal, 1374
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1406
 - ~DataResponseMarshaller, 1407
 - DataResponseMarshaller, 1407
 - createObject, 1407
 - getDataStructureType, 1407
 - looseMarshal, 1407
 - looseUnmarshal, 1408
 - tightMarshal1, 1408
 - tightMarshal2, 1408
 - tightUnmarshal, 1409
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1505

- ~DestinationInfoMarshaller, 1506
- createObject, 1506
- DestinationInfoMarshaller, 1506
- getDataStructureType, 1506
- looseMarshal, 1506
- looseUnmarshal, 1506
- tightMarshal1, 1507
- tightMarshal2, 1507
- tightUnmarshal, 1508
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1527
 - ~DiscoveryEventMarshaller, 1528
 - createObject, 1528
 - DiscoveryEventMarshaller, 1528
 - getDataStructureType, 1528
 - looseMarshal, 1528
 - looseUnmarshal, 1528
 - tightMarshal1, 1529
 - tightMarshal2, 1529
 - tightUnmarshal, 1530
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1601
 - ~ExceptionResponseMarshaller, 1602
 - createObject, 1602
 - ExceptionResponseMarshaller, 1602
 - getDataStructureType, 1602
 - looseMarshal, 1602
 - looseUnmarshal, 1603
 - tightMarshal1, 1603
 - tightMarshal2, 1603
 - tightUnmarshal, 1604
- activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1695
 - ~FlushCommandMarshaller, 1696
 - createObject, 1696
 - FlushCommandMarshaller, 1696
 - getDataStructureType, 1696
 - looseMarshal, 1696
 - looseUnmarshal, 1696
 - tightMarshal1, 1697
 - tightMarshal2, 1697
 - tightUnmarshal, 1698
- activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1796
 - ~IntegerResponseMarshaller, 1797
 - createObject, 1797
 - getDataStructureType, 1797
 - IntegerResponseMarshaller, 1797
 - looseMarshal, 1797
 - looseUnmarshal, 1798
 - tightMarshal1, 1798
 - tightMarshal2, 1798
 - tightUnmarshal, 1799
- activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 1854
 - ~JournalQueueAckMarshaller, 1855
 - createObject, 1855
 - getDataStructureType, 1855
 - JournalQueueAckMarshaller, 1855
 - looseMarshal, 1855
 - looseUnmarshal, 1855
 - tightMarshal1, 1856
 - tightMarshal2, 1856
 - tightUnmarshal, 1857
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 1879
 - ~JournalTopicAckMarshaller, 1880
 - createObject, 1880
 - getDataStructureType, 1880
 - JournalTopicAckMarshaller, 1880
 - looseMarshal, 1880
 - looseUnmarshal, 1880
 - tightMarshal1, 1881
 - tightMarshal2, 1881
 - tightUnmarshal, 1882
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 1902
 - ~JournalTraceMarshaller, 1903
 - createObject, 1903
 - getDataStructureType, 1903
 - JournalTraceMarshaller, 1903
 - looseMarshal, 1903
 - looseUnmarshal, 1903
 - tightMarshal1, 1904
 - tightMarshal2, 1904
 - tightUnmarshal, 1905
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1926
 - ~JournalTransactionMarshaller, 1927
 - createObject, 1927
 - getDataStructureType, 1927
 - JournalTransactionMarshaller, 1927
 - looseMarshal, 1927
 - looseUnmarshal, 1927
 - tightMarshal1, 1928
 - tightMarshal2, 1928
 - tightUnmarshal, 1929
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 1937
 - ~KeepAliveInfoMarshaller, 1938
 - createObject, 1938
 - getDataStructureType, 1938
 - KeepAliveInfoMarshaller, 1938
 - looseMarshal, 1938
 - looseUnmarshal, 1938
 - tightMarshal1, 1939
 - tightMarshal2, 1939

- tightUnmarshal, 1940
- activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 1977
 - ~LastPartialCommandMarshaller, 1978
 - createObject, 1978
 - getDataStructureType, 1978
 - LastPartialCommandMarshaller, 1978
 - looseMarshal, 1978
 - looseUnmarshal, 1979
 - tightMarshal1, 1979
 - tightMarshal2, 1979
 - tightUnmarshal, 1980
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2013
 - ~LocalTransactionIdMarshaller, 2014
 - createObject, 2014
 - getDataStructureType, 2014
 - LocalTransactionIdMarshaller, 2014
 - looseMarshal, 2014
 - looseUnmarshal, 2014
 - tightMarshal1, 2015
 - tightMarshal2, 2015
 - tightUnmarshal, 2016
- activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2125
 - ~MarshallerFactory, 2125
 - configure, 2125
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2198
 - ~MessageAckMarshaller, 2199
 - createObject, 2199
 - getDataStructureType, 2199
 - looseMarshal, 2199
 - looseUnmarshal, 2199
 - MessageAckMarshaller, 2199
 - tightMarshal1, 2200
 - tightMarshal2, 2200
 - tightUnmarshal, 2201
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2247
 - ~MessageDispatchMarshaller, 2248
 - createObject, 2248
 - getDataStructureType, 2248
 - looseMarshal, 2248
 - looseUnmarshal, 2248
 - MessageDispatchMarshaller, 2248
 - tightMarshal1, 2249
 - tightMarshal2, 2249
 - tightUnmarshal, 2250
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2273
 - ~MessageDispatchNotificationMarshaller, 2274
 - createObject, 2274
- getDataStructureType, 2274
- looseUnmarshal, 2274
- MessageDispatchNotificationMarshaller, 2274
- tightMarshal1, 2275
- tightMarshal2, 2275
- tightUnmarshal, 2276
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2296
 - ~MessageIdMarshaller, 2297
 - createObject, 2297
 - getDataStructureType, 2297
 - looseMarshal, 2297
 - looseUnmarshal, 2297
 - MessageIdMarshaller, 2297
 - tightMarshal1, 2298
 - tightMarshal2, 2298
 - tightUnmarshal, 2299
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2320
 - ~MessageMarshaller, 2321
 - looseMarshal, 2321
 - looseUnmarshal, 2321
 - MessageMarshaller, 2321
 - tightMarshal1, 2322
 - tightMarshal2, 2322
 - tightUnmarshal, 2323
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2363
 - ~MessagePullMarshaller, 2364
 - createObject, 2364
 - getDataStructureType, 2364
 - looseMarshal, 2364
 - looseUnmarshal, 2364
 - MessagePullMarshaller, 2364
 - tightMarshal1, 2365
 - tightMarshal2, 2365
- activemq::wireformat::openwire::marshal::v5::MessageUnmarshal, 2366
 - ~MessageUnmarshal, 2367
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2405
 - ~NetworkBridgeFilterMarshaller, 2406
 - createObject, 2406
 - getDataStructureType, 2406
 - looseMarshal, 2406
 - looseUnmarshal, 2406
 - NetworkBridgeFilterMarshaller, 2406
 - tightMarshal1, 2407
 - tightMarshal2, 2407
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2493
 - ~PartialCommandMarshaller, 2494
 - createObject, 2494

- getDataStructureType, 2494
 - looseMarshal, 2494
 - looseUnmarshal, 2495
 - PartialCommandMarshaller, 2494
 - tightMarshal1, 2495
 - tightMarshal2, 2495
 - tightUnmarshal, 2496
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2595
 - ~ProducerAckMarshaller, 2596
 - createObject, 2596
 - getDataStructureType, 2596
 - looseMarshal, 2596
 - looseUnmarshal, 2596
 - ProducerAckMarshaller, 2596
 - tightMarshal1, 2597
 - tightMarshal2, 2597
 - tightUnmarshal, 2598
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2619
 - ~ProducerIdMarshaller, 2620
 - createObject, 2620
 - getDataStructureType, 2620
 - looseMarshal, 2620
 - looseUnmarshal, 2620
 - ProducerIdMarshaller, 2620
 - tightMarshal1, 2621
 - tightMarshal2, 2621
 - tightUnmarshal, 2622
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2640
 - ~ProducerInfoMarshaller, 2641
 - createObject, 2641
 - getDataStructureType, 2641
 - looseMarshal, 2641
 - looseUnmarshal, 2641
 - ProducerInfoMarshaller, 2641
 - tightMarshal1, 2642
 - tightMarshal2, 2642
 - tightUnmarshal, 2643
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2723
 - ~RemoveInfoMarshaller, 2724
 - createObject, 2724
 - getDataStructureType, 2724
 - looseMarshal, 2724
 - looseUnmarshal, 2724
 - RemoveInfoMarshaller, 2724
 - tightMarshal1, 2725
 - tightMarshal2, 2725
 - tightUnmarshal, 2726
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2732
 - ~RemoveSubscriptionInfoMarshaller, 2733
- createObject, 2733
- getDataStructureType, 2733
- looseMarshal, 2733
- looseUnmarshal, 2733
- RemoveSubscriptionInfoMarshaller, 2733
- tightMarshal1, 2734
- tightMarshal2, 2734
- ~AckMarshaller, 2735
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2764
 - ~ReplayCommandMarshaller, 2765
 - createObject, 2765
 - getDataStructureType, 2765
 - looseMarshal, 2765
 - looseUnmarshal, 2765
 - ReplayCommandMarshaller, 2765
 - tightMarshal1, 2766
 - tightMarshal2, 2766
 - ~IdMarshaller, 2767
 - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2801
 - ~ResponseMarshaller, 2802
 - createObject, 2802
 - getDataStructureType, 2802
 - looseMarshal, 2802
 - looseUnmarshal, 2803
 - ResponseMarshaller, 2802
 - tightMarshal1, 2803
 - tightMarshal2, 2804
 - ~InfoMarshaller, 2804
 - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 2857
 - ~SessionIdMarshaller, 2858
 - createObject, 2858
 - getDataStructureType, 2858
 - looseMarshal, 2858
 - looseUnmarshal, 2858
 - SessionIdMarshaller, 2858
 - tightMarshal1, 2859
 - tightMarshal2, 2859
 - ~InfoMarshaller, 2860
 - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2893
 - ~SessionInfoMarshaller, 2894
 - createObject, 2894
 - getDataStructureType, 2894
 - looseMarshal, 2894
 - looseUnmarshal, 2894
 - SessionInfoMarshaller, 2894
 - tightMarshal1, 2895
 - tightMarshal2, 2895
 - ~ShutdownInfoMarshaller, 2896
 - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2953

- ~ShutdownInfoMarshaller, 2954
- createObject, 2954
- getDataStructureType, 2954
- looseMarshal, 2954
- looseUnmarshal, 2954
- ShutdownInfoMarshaller, 2954
- tightMarshal1, 2955
- tightMarshal2, 2955
- tightUnmarshal, 2956
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3114
 - ~SubscriptionInfoMarshaller, 3115
 - createObject, 3115
 - getDataStructureType, 3115
 - looseMarshal, 3115
 - looseUnmarshal, 3115
 - SubscriptionInfoMarshaller, 3115
 - tightMarshal1, 3116
 - tightMarshal2, 3116
 - tightUnmarshal, 3117
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3232
 - ~TransactionIdMarshaller, 3233
 - looseMarshal, 3233
 - looseUnmarshal, 3233
 - tightMarshal1, 3234
 - tightMarshal2, 3234
 - tightUnmarshal, 3235
 - TransactionIdMarshaller, 3233
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3257
 - ~TransactionInfoMarshaller, 3258
 - createObject, 3258
 - getDataStructureType, 3258
 - looseMarshal, 3258
 - looseUnmarshal, 3258
 - tightMarshal1, 3259
 - tightMarshal2, 3259
 - tightUnmarshal, 3260
 - TransactionInfoMarshaller, 3258
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3383
 - ~WireFormatInfoMarshaller, 3384
 - createObject, 3384
 - getDataStructureType, 3384
 - looseMarshal, 3384
 - looseUnmarshal, 3384
 - tightMarshal1, 3385
 - tightMarshal2, 3385
 - tightUnmarshal, 3386
 - WireFormatInfoMarshaller, 3384
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3431
 - ~XATransactionIdMarshaller, 3432
 - createObject, 3432
 - getDataStructureType, 3432
 - looseMarshal, 3432
 - looseUnmarshal, 3432
 - tightMarshal1, 3433
 - tightMarshal2, 3433
 - tightUnmarshal, 3434
 - XATransactionIdMarshaller, 3432
- activemq::wireformat::openwire::OpenWireFormat, 2451
 - ~OpenWireFormat, 2451
 - addMarshaller, 2451
 - createNegotiator, 2451
 - DEFAULT_VERSION, 2459
 - destroyMarshallers, 2451
 - doUnmarshal, 2451
 - getCacheSize, 2452
 - getMaxInactivityDuration, 2452
 - getMaxInactivityDurationInitialDelay, 2452
 - getOpenWireFormatInfo, 2452
 - getVersion, 2453
 - hasNegotiator, 2453
 - inReceive, 2453
 - isCacheEnabled, 2453
 - isSizePrefixDisabled, 2453
 - isStackTraceEnabled, 2454
 - isTcpNoDelayEnabled, 2454
 - isTightEncodingEnabled, 2454
 - looseMarshalNestedObject, 2454
 - looseUnmarshalNestedObject, 2455
 - marshal, 2455
 - NULL_TYPE, 2459
 - OpenWireFormat, 2451
 - renegotiateWireFormat, 2455
 - setCacheEnabled, 2456
 - setCacheSize, 2456
 - setMaxInactivityDuration, 2456
 - setMaxInactivityDurationInitialDelay, 2456
 - setSizePrefixDisabled, 2457
 - setStackTraceEnabled, 2457
 - setTcpNoDelayEnabled, 2457
 - setTightEncodingEnabled, 2457
 - setVersion, 2457
 - tightMarshalNestedObject1, 2458
 - tightMarshalNestedObject2, 2458
 - tightUnmarshalNestedObject, 2458
 - unmarshal, 2459
- activemq::wireformat::openwire::OpenWireFormatFactory, 2460
 - ~OpenWireFormatFactory, 2460
 - createWireFormat, 2460

- OpenWireFormatFactory, 2460
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2462
 - ~OpenWireFormatNegotiator, 2463
 - close, 2463
 - onCommand, 2463
 - oneway, 2463
 - onTransportException, 2464
 - OpenWireFormatNegotiator, 2462
 - request, 2464
 - start, 2465
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2466
 - ~OpenWireResponseBuilder, 2466
 - buildIncomingCommands, 2466
 - buildResponse, 2466
 - OpenWireResponseBuilder, 2466
- activemq::wireformat::openwire::utils, 110
- activemq::wireformat::openwire::utils::BooleanStream, 739
 - ~BooleanStream, 740
 - BooleanStream, 740
 - clear, 740
 - marshal, 740
 - marshalledSize, 740
 - readBoolean, 740
 - unmarshal, 740
 - writeBoolean, 741
- activemq::wireformat::openwire::utils::HexTable, 1715
 - ~HexTable, 1715
 - HexTable, 1715
 - operator[], 1715
 - size, 1716
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2339
 - ~MessagePropertyInterceptor, 2340
 - getBooleanProperty, 2341
 - getByteProperty, 2341
 - getDoubleProperty, 2341
 - getFloatProperty, 2341
 - getIntProperty, 2342
 - getLongProperty, 2342
 - getShortProperty, 2342
 - getStringProperty, 2342
 - MessagePropertyInterceptor, 2340
 - setBooleanProperty, 2343
 - setByteProperty, 2343
 - setDoubleProperty, 2343
 - setFloatProperty, 2343
 - setIntProperty, 2344
 - setLongProperty, 2344
 - setShortProperty, 2344
 - setStringProperty, 2344
- activemq::wireformat::openwire::utils::OpenwireStringSupport, 2468
 - ~OpenwireStringSupport, 2468
 - OpenwireStringSupport, 2468
 - readString, 2468
 - writeString, 2469
- activemq::wireformat::stomp, 111
- activemq::wireformat::stomp::StompCommandConstants, 3057
 - ABORT, 3059
 - ACK, 3059
 - ACK_AUTO, 3059
 - ACK_CLIENT, 3059
 - ACK_INDIVIDUAL, 3059
 - BEGIN, 3059
 - BYTES, 3059
 - COMMIT, 3059
 - CONNECT, 3059
 - CONNECTED, 3059
 - DISCONNECT, 3059
 - ERROR_CMD, 3059
 - HEADER_ACK, 3059
 - HEADER_CLIENT_ID, 3059
 - HEADER_CONSUMERPRIORITY, 3059
 - HEADER_CONTENTLENGTH, 3059
 - HEADER_CORRELATIONID, 3059
 - HEADER_DESTINATION, 3059
 - HEADER_DISPATCH_ASYNC, 3059
 - HEADER_EXCLUSIVE, 3059
 - HEADER_EXPIRES, 3059
 - HEADER_ID, 3059
 - HEADER_JMSPRIORITY, 3059
 - HEADER_LOGIN, 3059
 - HEADER_MAXPENDINGMSGLIMIT, 3059
 - HEADER_MESSAGE, 3059
 - HEADER_MESSAGEID, 3059
 - HEADER_NOLOCAL, 3059
 - HEADER_OLDSUBSCRIPTIONNAME, 3059
 - HEADER_PASSWORD, 3059
 - HEADER_PERSISTENT, 3059
 - HEADER_PREFETCHSIZE, 3059
 - HEADER_RECEIPT_REQUIRED, 3059
 - HEADER_RECEIPTID, 3059
 - HEADER_REDELIVERED, 3059
 - HEADER_REDELIVERYCOUNT, 3059
 - HEADER_REPLYTO, 3059
 - HEADER_REQUESTID, 3059
 - HEADER_RESPONSEID, 3059
 - HEADER_RETROACTIVE, 3059
 - HEADER_SELECTOR, 3059
 - HEADER_SESSIONID, 3059
 - HEADER_SUBSCRIPTION, 3059

- HEADER_SUBSCRIPTIONNAME, 3059
- HEADER_TIMESTAMP, 3059
- HEADER_TRANSACTIONID, 3059
- HEADER_TRANSFORMATION, 3059
- HEADER_TRANSFORMATION_ -
ERROR, 3059
- HEADER_TYPE, 3059
- MESSAGE, 3059
- QUEUE_PREFIX, 3059
- RECEIPT, 3059
- SEND, 3059
- SUBSCRIBE, 3059
- TEMPQUEUE_PREFIX, 3059
- TEMPTOPIC_PREFIX, 3059
- TEXT, 3059
- TOPIC_PREFIX, 3059
- UNSUBSCRIBE, 3059
- activemq::wireformat::stomp::StompFrame, 3061
 - ~StompFrame, 3062
 - clone, 3062
 - copy, 3062
 - fromStream, 3062
 - getBody, 3063
 - getBodyLength, 3063
 - getCommand, 3063
 - getProperties, 3063
 - getProperty, 3064
 - hasProperty, 3064
 - removeProperty, 3064
 - setBody, 3064
 - setCommand, 3064
 - setProperty, 3065
 - StompFrame, 3062
 - toStream, 3065
- activemq::wireformat::stomp::StompHelper, 3066
 - ~StompHelper, 3067
 - convertConsumerId, 3067
 - convertDestination, 3067, 3068
 - convertMessageId, 3068
 - convertProducerId, 3068, 3069
 - convertProperties, 3069
 - convertTransactionId, 3069, 3070
 - StompHelper, 3067
- activemq::wireformat::stomp::StompWireFormat, 3071
 - ~StompWireFormat, 3072
 - createNegotiator, 3072
 - getVersion, 3072
 - hasNegotiator, 3072
 - inReceive, 3072
 - marshal, 3073
 - setVersion, 3073
 - StompWireFormat, 3072
 - unmarshal, 3073
- activemq::wireformat::stomp::StompWireFormatFactory, 3075
 - ~StompWireFormatFactory, 3075
 - createWireFormat, 3075
 - StompWireFormatFactory, 3075
- activemq::wireformat::WireFormat, 3363
 - ~WireFormat, 3364
 - createNegotiator, 3364
 - getVersion, 3364
 - hasNegotiator, 3364
 - inReceive, 3365
 - marshal, 3365
 - setVersion, 3365
 - unmarshal, 3365
- activemq::wireformat::WireFormatFactory, 3367
 - ~WireFormatFactory, 3367
 - createWireFormat, 3367
- activemq::wireformat::WireFormatNegotiator, 3399
 - ~WireFormatNegotiator, 3399
 - WireFormatNegotiator, 3399
- activemq::wireformat::WireFormatRegistry, 3400
 - ~WireFormatRegistry, 3401
 - findFactory, 3401
 - getInstance, 3401
 - getWireFormatNames, 3401
 - registerFactory, 3401
 - unregisterFactory, 3402
- ActiveMQBlobMessage
 - activemq::commands::ActiveMQBlobMessage, 173
- ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 182
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobM, 194
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobM, 178
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobM, 186
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobM, 190
- ActiveMQBytesMessage
 - activemq::commands::ActiveMQBytesMessage, 200
- ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesM, 218
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesM, 230

- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 214
- activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 222
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 226
- ActiveMQConnection
 - activemq::core::ActiveMQConnection, 235
- ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnectionFactory, 245
- ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnectionMetaData, 250
- ActiveMQConnectionSupport
 - activemq::core::ActiveMQConnectionSupport, 254
- ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 265
- ActiveMQCPP
 - activemq::library::ActiveMQCPP, 272
- ActiveMQDestination
 - activemq::commands::ActiveMQDestination, 276
- ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 290
 - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 302
 - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 286
 - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 294
 - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 298
- ActiveMQException
 - activemq::exceptions::ActiveMQException, 305, 306
- ActiveMQMapMessage
 - activemq::commands::ActiveMQMapMessage, 311
- ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 327
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 339
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 323
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 331
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 335
- ActiveMQMessage
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessage, 342
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMessage, 362
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMessage, 346
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMessage, 354
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMessage, 358
- ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 368
- ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 384
- ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 391
 - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 403
 - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 387
 - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 395
 - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 399
- ActiveMQProducer
 - activemq::core::ActiveMQProducer, 407
- ActiveMQQueue
 - activemq::core::ActiveMQQueue, 420
- ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 428
 - activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 440
 - activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 424
 - activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 432
 - activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 436
- ActiveMQSession
 - activemq::core::ActiveMQSession, 447
- ActiveMQSessionExecutor
 - activemq::core::ActiveMQSession, 459
- ActiveMQStreamMessage
 - activemq::core::ActiveMQStreamMessage, 461

- activemq::commands::ActiveMQStreamMessage, 566
- 467
- ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 484
 - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 496
 - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 480
 - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 488
 - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 492
- ActiveMQTempDestination
 - activemq::commands::ActiveMQTempDestination, 500
- ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 508
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 520
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 504
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 512
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 516
- ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 524
- ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 533
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 545
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 529
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 537
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 541
- ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 549
- ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 558
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 570
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 554
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 562
- ActiveMQTextMessage
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 574
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 583
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 595
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 579
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 587
- ActiveMQTopic
 - activemq::commands::ActiveMQTopic, 599
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 607
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 619
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 603
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 611
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 615
- ActiveMQTransactionContext
 - activemq::core::ActiveMQTransactionContext, 623
- add
 - activemq::transport::failover::CloseTransportsTask, 1022
 - activemq::transport::failover::FailoverTransport, 1014
 - decaf::util::AbstractCollection, 150
 - decaf::util::AbstractQueue, 163
 - decaf::util::Collection, 1056
 - decaf::util::List, 1985
 - decaf::util::ListIterator, 1991
 - decaf::util::PriorityQueue, 2574
 - decaf::util::StlList, 3022
 - decaf::util::StlSet, 3053
- addAll
 - decaf::util::AbstractCollection, 150
 - decaf::util::AbstractQueue, 163
 - decaf::util::Collection, 1056
 - decaf::util::List, 1985
- addAndGet
 - java.lang.AtomicInteger, 634

- addCommand
 - activemq::state::TransactionState, 3266
- addConnection
 - activemq::cmsutil::ResourceLifecycleManager, 2779
- addConsumer
 - activemq::state::SessionState, 2904
- addDestination
 - activemq::cmsutil::ResourceLifecycleManager, 2779
- addDispatcher
 - activemq::core::ActiveMQConnection, 236
- addHandler
 - decaf::util::logging::Logger, 2030
- additionalPredicate
 - activemq::commands::ConsumerInfo, 1307
- addMarshaller
 - activemq::wireformat::openwire::OpenWireFormat, 2451
- addMessageConsumer
 - activemq::cmsutil::ResourceLifecycleManager, 2779
- addMessageProducer
 - activemq::cmsutil::ResourceLifecycleManager, 2779
- addProducer
 - activemq::core::ActiveMQConnection, 236
 - activemq::state::SessionState, 2904
- addPropertyChangeListener
 - decaf::util::logging::LogManager, 2047
- addSession
 - activemq::cmsutil::ResourceLifecycleManager, 2779
 - activemq::state::ConnectionState, 1242
- addSynchronization
 - activemq::core::ActiveMQTransactionContext, 623
- addTask
 - activemq::threads::CompositeTaskRunner, 1093
- addTempDestination
 - activemq::state::ConnectionState, 1242
- addTransactionState
 - activemq::state::ConnectionState, 1242
- addTransportListener
 - activemq::core::ActiveMQConnection, 236
- addURI
 - activemq::transport::CompositeTransport, 1095
 - activemq::transport::failover::FailoverTransport, 1614
 - activemq::transport::failover::URIPool, 3330
- addURIs
 - activemq::transport::failover::URIPool, 3330
- adjustMinimum
 - decaf::internal::util::TimerTaskHeap, 3203
- advisory
 - activemq::commands::ActiveMQDestination, 283
- ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 283
- after
 - decaf::util::Date, 1468
- afterCommit
 - activemq::core::Synchronization, 3137
- afterMarshal
 - activemq::commands::BaseDataStructure, 716
- activemq::wireformat::MarshalAware, 2118
 - afterMessageIsConsumed
 - activemq::core::ActiveMQConsumer, 265
 - afterRollback
 - activemq::core::Synchronization, 3137
 - afterUnmarshal
 - activemq::commands::BaseDataStructure, 716
 - activemq::commands::Message, 2149
 - activemq::commands::WireFormatInfo, 3371
 - activemq::wireformat::MarshalAware, 2119
- allocate
 - decaf::nio::ByteBuffer, 918
 - decaf::nio::CharBuffer, 1001
 - decaf::nio::DoubleBuffer, 1558
 - decaf::nio::FloatBuffer, 1667
 - decaf::nio::IntBuffer, 1753
 - decaf::nio::LongBuffer, 2081
 - decaf::nio::ShortBuffer, 2925
- AMQ_CATCH_ALL_THROW - CMSEXCEPTION
 - CMSEExceptionSupport.h, 3586
- AMQ_CATCH_EXCEPTION_CONVERT
 - activemq/exceptions/ExceptionDefines.h, 3533
- AMQ_CATCH_NOTHROW
 - activemq/exceptions/ExceptionDefines.h, 3533
- AMQ_CATCH_RETHROW
 - activemq/exceptions/ExceptionDefines.h, 3534
- AMQ_CATCHALL_NOTHROW
 - activemq/exceptions/ExceptionDefines.h, 3534
- AMQ_CATCHALL_THROW

- activemq/exceptions/ExceptionDefines.h, 3534
- AMQCPP_API
 - activemq/util/Config.h, 3589
- ANY_CHILD
 - activemq::commands::ActiveMQDestination::DestinationFilter, 1483
- ANY_DESCENDENT
 - activemq::commands::ActiveMQDestination::DestinationFilter, 1483
- append
 - decaf::lang::Appendable, 626, 627
 - decaf::nio::CharBuffer, 1001, 1002
- AprPool
 - decaf::internal::nio::AprPool, 628
- array
 - decaf::internal::nio::ByteBuffer, 876
 - decaf::internal::nio::CharArrayBuffer, 993
 - decaf::internal::nio::DoubleArrayBuffer, 1550
 - decaf::internal::nio::FloatArrayBuffer, 1660
 - decaf::internal::nio::IntArrayBuffer, 1746
 - decaf::internal::nio::LongArrayBuffer, 2073
 - decaf::internal::nio::ShortArrayBuffer, 2917
 - decaf::nio::ByteBuffer, 918
 - decaf::nio::CharBuffer, 1003
 - decaf::nio::DoubleBuffer, 1558
 - decaf::nio::FloatBuffer, 1667
 - decaf::nio::IntBuffer, 1753
 - decaf::nio::LongBuffer, 2081
 - decaf::nio::ShortBuffer, 2925
- arrayOffset
 - decaf::internal::nio::ByteBuffer, 876
 - decaf::internal::nio::CharArrayBuffer, 993
 - decaf::internal::nio::DoubleArrayBuffer, 1551
 - decaf::internal::nio::FloatArrayBuffer, 1661
 - decaf::internal::nio::IntArrayBuffer, 1747
 - decaf::internal::nio::LongArrayBuffer, 2074
 - decaf::internal::nio::ShortArrayBuffer, 2918
 - decaf::nio::ByteBuffer, 919
 - decaf::nio::CharBuffer, 1003
 - decaf::nio::DoubleBuffer, 1559
 - decaf::nio::FloatBuffer, 1668
 - decaf::nio::IntBuffer, 1754
 - decaf::nio::LongBuffer, 2082
 - decaf::nio::ShortBuffer, 2926
- arrival
 - activemq::commands::Message, 2161
- asCharBuffer
 - decaf::internal::nio::ByteBuffer, 876
 - decaf::nio::ByteBuffer, 919
- asDoubleBuffer
 - decaf::internal::nio::ByteBuffer, 877
 - decaf::nio::ByteBuffer, 919
- asFloatBuffer
 - decaf::internal::nio::ByteBuffer, 877
 - decaf::internal::nio::ByteBuffer, 920
- asIntBuffer
 - decaf::internal::nio::ByteBuffer, 877
 - decaf::internal::nio::ByteBuffer, 920
- asLongBuffer
 - decaf::internal::nio::ByteBuffer, 878
 - decaf::nio::ByteBuffer, 920
- asReadOnlyBuffer
 - decaf::internal::nio::ByteBuffer, 878
 - decaf::internal::nio::CharArrayBuffer, 993
 - decaf::internal::nio::DoubleArrayBuffer, 1551
 - decaf::internal::nio::FloatArrayBuffer, 1661
 - decaf::internal::nio::IntArrayBuffer, 1747
 - decaf::internal::nio::LongArrayBuffer, 2074
 - decaf::internal::nio::ShortArrayBuffer, 2918
 - decaf::nio::ByteBuffer, 921
 - decaf::nio::CharBuffer, 1003
 - decaf::nio::DoubleBuffer, 1559
 - decaf::nio::FloatBuffer, 1668
 - decaf::nio::IntBuffer, 1754
 - decaf::nio::LongBuffer, 2082
 - decaf::nio::ShortBuffer, 2926
- asShortBuffer
 - decaf::internal::nio::ByteBuffer, 878
 - decaf::nio::ByteBuffer, 921
- AsyncSignalReadErrorTask
 - activemq::transport::inactivity::InactivityMonitor, 1736
- AsyncWriteTask
 - activemq::transport::inactivity::InactivityMonitor, 1736
- AtomicBoolean
 - decaf::util::concurrent::atomic::AtomicBoolean, 630
- AtomicInteger
 - decaf::util::concurrent::atomic::AtomicInteger, 634
- AtomicRefCounter
 - decaf::lang::AtomicRefCounter, 639
- AtomicReference
 - decaf::util::concurrent::atomic::AtomicReference, 642
- AUTO_ACKNOWLEDGE
 - cms::Session, 2842
- available
 - decaf::internal::nio::StandardInputStream, 3001

- decaf::io::BlockingByteArrayInputStream, 726
- decaf::io::BufferedInputStream, 810
- decaf::io::ByteArrayInputStream, 894
- decaf::io::FilterInputStream, 1633
- decaf::io::InputStream, 1741
- decaf::net::SocketInputStream, 2979
- availablePermits
 - decaf::util::concurrent::Semaphore, 2831
- availableProcessors
 - decaf::lang::System, 3145
- await
 - decaf::util::concurrent::CountDownLatch, 1354
 - decaf::util::concurrent::locks::Condition, 1120
- awaitNanos
 - decaf::util::concurrent::locks::Condition, 1121
- awaitTermination
 - decaf::util::concurrent::ExecutorService, 1611
- awaitUninterruptibly
 - decaf::util::concurrent::locks::Condition, 1122
- awaitUntil
 - decaf::util::concurrent::locks::Condition, 1123
- back
 - decaf::util::StlQueue, 3045
- BackupTransport
 - activemq::transport::failover::BackupTransport, 644
 - activemq::transport::failover::BackupTransportPool, 649
- BackupTransportPool
 - activemq::transport::failover::BackupTransportPool, 648
- BaseCommand
 - activemq::commands::BaseCommand, 651
- BaseCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 665
 - activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 686
 - activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 658
 - activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 672
 - activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 679
- before
 - decaf::util::Date, 1468
- beforeEnd
 - activemq::core::Synchronization, 3137
- beforeMarshal
 - activemq::commands::ActiveMQMapMessage, 311
 - activemq::commands::ActiveMQTextMessage, 574
 - activemq::commands::BaseDataStructure, 716
 - activemq::commands::Message, 2149
 - activemq::commands::WireFormatInfo, 3372
 - activemq::wireformat::MarshalAware, 2119
- beforeMessageIsConsumed
 - activemq::core::ActiveMQConsumer, 266
- beforeUnmarshal
 - activemq::commands::BaseDataStructure, 716
 - activemq::wireformat::MarshalAware, 2119
- BEGIN
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- begin
 - activemq::core::ActiveMQTransactionContext, 623
- BIG_STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- BINARY_MIME_TYPE
 - activemq::commands::ActiveMQBlobMessage, 176
- bind
 - decaf::net::ServerSocket, 2838
- BindException
 - decaf::net::BindException, 721, 722
- bitCount
 - decaf::lang::Integer, 1765
 - decaf::lang::Long, 2060
- BlockingByteArrayInputStream
 - decaf::io::BlockingByteArrayInputStream, 725
- Boolean
 - decaf::lang::Boolean, 733
- BOOLEAN_TYPE
 - activemq::util::PrimitiveValueNode, 2558
- BooleanExpression
 - activemq::commands::BooleanExpression, 737
- BooleanStream
 - activemq::wireformat::openwire::utils::BooleanStream, 740
- booleanValue
 - decaf::lang::Boolean, 733
- boolValue

- activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2552
- branchQualifier
 - activemq::commands::XATransactionId, 3414
- BrokenBarrierException
 - decaf::util::concurrent::BrokenBarrierException, 742, 743
- BrokerError
 - activemq::commands::BrokerError, 746
- BrokerException
 - activemq::exceptions::BrokerException, 749
- BrokerId
 - activemq::commands::BrokerId, 752
- brokerId
 - activemq::commands::BrokerInfo, 782
- BrokerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 760
 - activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 772
 - activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 756
 - activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 764
 - activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 768
- BrokerInfo
 - activemq::commands::BrokerInfo, 777
- BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 788
 - activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 800
 - activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 784
 - activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 792
 - activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 796
- brokerInTime
 - activemq::commands::Message, 2161
- brokerMasterConnector
 - activemq::commands::ConnectionInfo, 1216
- brokerName
 - activemq::commands::BrokerInfo, 782
 - activemq::commands::DiscoveryEvent, 1514
- brokerOutTime
 - activemq::commands::Message, 2161
- brokerPath
 - activemq::commands::ConnectionInfo, 1216
 - activemq::commands::ConsumerInfo, 1307
 - activemq::commands::DestinationInfo, 1488
 - activemq::commands::Message, 2161
 - activemq::commands::ProducerInfo, 2635
 - brokerSequenceId
 - activemq::commands::MessageId, 2283
 - brokerUploadUrl
 - activemq::commands::BrokerInfo, 782
 - brokerURL
 - activemq::commands::BrokerInfo, 782
 - browser
 - activemq::commands::ConsumerInfo, 1307
 - Buffer
 - decaf::nio::Buffer, 805
 - buffer
 - activemq::transport::mock::DataOutputStream, 1394
 - BufferedInputStream
 - activemq::transport::mock::BufferedInputStream, 810
 - BufferedOutputStream
 - activemq::transport::mock::BufferedOutputStream, 814, 815
 - BufferedReader
 - activemq::transport::mock::BufferedReader, 818
 - BufferOverflowException
 - activemq::transport::mock::BufferOverflowException, 834, 835
 - BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 837, 838
 - BrokerInfoManager
 - activemq::transport::mock::ResponseBuilder, 2785
 - BrokerInfoMarshaller
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2466
 - buildMessage
 - activemq::transport::mock::ResponseBuilder, 2785
 - buildResponse
 - activemq::transport::mock::ResponseBuilder, 2785
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2466
- Byte
 - decaf::lang::Byte, 841
- BYTE_ARRAY_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- BYTE_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 853–855
- ByteBuffer

- decaf::internal::nio::ByteBuffer, 874, 875
- ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 894
- ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 901
- ByteArrayPerspective
 - decaf::internal::nio::ByteArrayPerspective, 909–911
- byteArrayValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2552
- ByteBuffer
 - decaf::nio::ByteBuffer, 918
- BYTES
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- byteValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2552
 - decaf::lang::Byte, 842
 - decaf::lang::Character, 984
 - decaf::lang::Double, 1539
 - decaf::lang::Float, 1649
 - decaf::lang::Integer, 1765
 - decaf::lang::Long, 2060
 - decaf::lang::Number, 2432
 - decaf::lang::Short, 2908
- CachedConsumer
 - activemq::cmsutil::CachedConsumer, 954
- CachedProducer
 - activemq::cmsutil::CachedProducer, 958
- call
 - decaf::util::concurrent::Callable, 964
- cancel
 - decaf::util::concurrent::Future, 1702
 - decaf::util::Timer, 3190
 - decaf::util::TimerTask, 3200
- CancellationException
 - decaf::util::concurrent::CancellationException, 965, 966
- capacity
 - decaf::nio::Buffer, 805
- cause
 - decaf::lang::Exception, 1580
- ceil
 - decaf::lang::Math, 2129
- CertificateEncodingException
 - decaf::security::cert::CertificateEncodingException, 972, 973
- CertificateException
 - decaf::security::cert::CertificateException, 974, 975
- CertificateExpiredException
 - decaf::security::cert::CertificateExpiredException, 976, 977
- CertificateNotYetValidException
 - decaf::security::cert::CertificateNotYetValidException, 978, 979
- CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 980, 981
- CHAR_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- Character
 - decaf::lang::Character, 984
- CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 991, 992
- charAt
 - decaf::lang::CharSequence, 1014
- decaf::nio::CharBuffer, 1004
- CharBuffer
 - decaf::nio::CharBuffer, 1001
- charValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2552
- checkConnectionFactory
 - activemq::cmsutil::CmsAccessor, 1025
- checkDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 1029
- checkMapIsUnmarshalled
 - activemq::commands::ActiveMQMapMessage, 311
- checkResult
 - decaf::net::TcpSocket, 3154
- checkShutdown
 - activemq::state::ConnectionState, 1242
 - activemq::state::SessionState, 2904
 - activemq::state::TransactionState, 3266
- checkValidity
 - decaf::security::cert::X509Certificate, 3407
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2443
- ClassCastException
 - decaf::lang::exceptions::ClassCastException, 1016, 1017
- ClassName
 - activemq::commands::BrokerError::StackTraceElement, 2993
- cleanup
 - decaf::internal::AprPool, 628
- clear
 - activemq::core::ActiveMQSessionExecutor, 461

- activemq::core::MessageDispatchChannel, 2225
- activemq::util::ActiveMQProperties, 415
- activemq::util::PrimitiveValueNode, 2561
- activemq::wireformat::openwire::utils::BooleanStream, 1216
- 740
- cms::CMSProperties, 1037
- decaf::internal::util::ByteArrayAdapter, 856
- decaf::nio::Buffer, 805
- decaf::util::AbstractCollection, 151
- decaf::util::AbstractQueue, 164
- decaf::util::Collection, 1057
- decaf::util::concurrent::ConcurrentStlMap, 1106
- decaf::util::concurrent::SynchronousQueue, 3140
- decaf::util::Map, 2095
- decaf::util::PriorityQueue, 2574
- decaf::util::Properties, 2659
- decaf::util::StlList, 3023
- decaf::util::StlMap, 3034
- decaf::util::StlQueue, 3045
- decaf::util::StlSet, 3054
- clearBody
 - activemq::commands::ActiveMQBytesMessage, 200
 - activemq::commands::ActiveMQMapMessage, 311
 - activemq::commands::ActiveMQMessageTemplate, 368
 - activemq::commands::ActiveMQStreamMessage, 467
 - activemq::commands::ActiveMQTextMessage, 574
 - cms::Message, 2167
- clearMessagesInProgress
 - activemq::core::ActiveMQConsumer, 266
 - activemq::core::ActiveMQSession, 447
 - activemq::core::ActiveMQSessionExecutor, 461
- clearProperties
 - activemq::commands::ActiveMQMessageTemplate, 369
 - cms::Message, 2168
- CLIENT_ACKNOWLEDGE
 - cms::Session, 2842
- clientId
 - activemq::commands::ConnectionInfo, 1216
 - activemq::commands::JournalTopicAck, 1862
 - activemq::commands::RemoveSubscriptionInfo, 2731
 - activemq::commands::SubscriptionInfo, 3101
 - clientMaster
 - activemq::commands::ConnectionInfo,
 - clockSequence
 - decaf::util::UUID, 3358
 - clone
 - activemq::commands::ActiveMQBlobMessage, 173
 - activemq::commands::ActiveMQBytesMessage, 200
 - activemq::commands::ActiveMQMapMessage, 311
 - activemq::commands::ActiveMQMessage, 342
 - activemq::commands::ActiveMQObjectMessage, 384
 - activemq::commands::ActiveMQQueue, 420
 - activemq::commands::ActiveMQStreamMessage, 467
 - activemq::commands::ActiveMQTempQueue, 524
 - activemq::commands::ActiveMQTempTopic, 549
 - activemq::commands::ActiveMQTextMessage, 574
 - activemq::commands::ActiveMQTopic, 599
 - activemq::exceptions::ActiveMQException, 306
 - activemq::exceptions::BrokerException, 749
 - activemq::util::ActiveMQProperties, 415
 - activemq::wireformat::stomp::StompFrame, 3062
 - cms::BytesMessage, 941
 - cms::CMSProperties, 1037
 - cms::Destination, 1481
 - cms::Message, 2168
 - decaf::io::EOFException, 1573
 - decaf::io::InterruptedIOException, 1807
 - decaf::io::IOException, 1822
 - decaf::io::UnsupportedEncodingException, 3302
 - decaf::io::UTFDataFormatException, 3355
 - decaf::lang::Exception, 1577
 - decaf::lang::exceptions::ClassCastException, 1018
 - decaf::lang::exceptions::IllegalArgument Exception, 1722
 - decaf::lang::exceptions::IllegalMonitorStateException, 1725

- decaf::lang::exceptions::IllegalStateException, 1728
- decaf::lang::exceptions::IllegalThreadStateException, 1732
- decaf::lang::exceptions::IndexOutOfBoundsException, 1739
- decaf::lang::exceptions::InterruptedException, 1804
- decaf::lang::exceptions::InvalidStateException, 1819
- decaf::lang::exceptions::NoSuchElementException, 2425
- decaf::lang::exceptions::NullPointerException, 2431
- decaf::lang::exceptions::NumberFormatException, 2437
- decaf::lang::exceptions::RuntimeException, 2821
- decaf::lang::exceptions::UnsupportedOperationException, 3305
- decaf::lang::Throwable, 3182
- decaf::net::BindException, 723
- decaf::net::ConnectException, 1130
- decaf::net::HttpRetryException, 1719
- decaf::net::MalformedURLException, 2093
- decaf::net::NoRouteToHostException, 2419
- decaf::net::PortUnreachableException, 2524
- decaf::net::ProtocolException, 2669
- decaf::net::SocketException, 2973
- decaf::net::SocketTimeoutException, 2992
- decaf::net::UnknownHostException, 3296
- decaf::net::UnknownServiceException, 3299
- decaf::net::URISyntaxException, 3337
- decaf::nio::BufferOverflowException, 836
- decaf::nio::BufferUnderflowException, 839
- decaf::nio::InvalidMarkException, 1815
- decaf::nio::ReadOnlyBufferException, 2687
- decaf::security::cert::CertificateEncodingException, 973
- decaf::security::cert::CertificateException, 975
- decaf::security::cert::CertificateExpiredException, 977
- decaf::security::cert::CertificateNotYetValidException, 979
- decaf::security::cert::CertificateParsingException, 981
- decaf::security::GeneralSecurityException, 1708
- decaf::security::InvalidKeyException, 1812
- decaf::security::KeyException, 1957
- decaf::security::NoSuchAlgorithmException, 2422
- decaf::security::NoSuchProviderException, 2428
- decaf::security::SignatureException, 2959
- decaf::util::concurrent::BrokenBarrierException, 744
- decaf::util::concurrent::CancellationException, 967
- decaf::util::concurrent::ExecutionException, 1607
- decaf::util::concurrent::RejectedExecutionException, 2701
- decaf::util::concurrent::TimeoutException, 3187
- decaf::util::Properties, 2659
- cloneDataStructure
- activemq::commands::ActiveMQBlobMessage, 173
- activemq::commands::ActiveMQBytesMessage, 200
- activemq::commands::ActiveMQDestination, 276
- activemq::commands::ActiveMQMapMessage, 311
- activemq::commands::ActiveMQMessage, 343
- activemq::commands::ActiveMQObjectMessage, 384
- activemq::commands::ActiveMQQueue, 420
- activemq::commands::ActiveMQStreamMessage, 467
- activemq::commands::ActiveMQTempDestination, 500
- activemq::commands::ActiveMQTempQueue, 524
- activemq::commands::ActiveMQTempTopic, 549
- activemq::commands::ActiveMQTextMessage, 575
- activemq::commands::ActiveMQTopic, 599
- activemq::commands::BooleanExpression, 737
- activemq::commands::BrokerError, 746
- activemq::commands::BrokerId, 752
- activemq::commands::BrokerInfo, 777
- activemq::commands::ConnectionControl, 1136
- activemq::commands::ConnectionError, 1161
- activemq::commands::ConnectionId, 1188
- activemq::commands::ConnectionInfo, 1212

- activemq::commands::ConsumerControl, 1251
- activemq::commands::ConsumerId, 1276
- activemq::commands::ConsumerInfo, 1302
- activemq::commands::ControlCommand, 1331
- activemq::commands::DataArrayResponse, 1357
- activemq::commands::DataResponse, 1396
- activemq::commands::DataStructure, 1461
- activemq::commands::DestinationInfo, 1485
- activemq::commands::DiscoveryEvent, 1512
- activemq::commands::ExceptionResponse, 1583
- activemq::commands::FlushCommand, 1677
- activemq::commands::IntegerResponse, 1778
- activemq::commands::JournalQueueAck, 1835
- activemq::commands::JournalTopicAck, 1859
- activemq::commands::JournalTrace, 1884
- activemq::commands::JournalTransaction, 1907
- activemq::commands::KeepAliveInfo, 1931
- activemq::commands::LastPartialCommand, 1959
- activemq::commands::LocalTransactionId, 1994
- activemq::commands::Message, 2149
- activemq::commands::MessageAck, 2189
- activemq::commands::MessageDispatch, 2220
- activemq::commands::MessageDispatchNotification, 2252
- activemq::commands::MessageId, 2280
- activemq::commands::MessagePull, 2347
- activemq::commands::NetworkBridgeFilter, 2394
- activemq::commands::PartialCommand, 2474
- activemq::commands::ProducerAck, 2580
- activemq::commands::ProducerId, 2607
- activemq::commands::ProducerInfo, 2632
- activemq::commands::RemoveInfo, 2704
- activemq::commands::RemoveSubscriptionInfo, 2728
- activemq::commands::ReplayCommand, 2753
- activemq::commands::Response, 2782
- activemq::commands::SessionId, 2854
- activemq::commands::SessionInfo, 2878
- activemq::commands::ShutdownInfo, 2935
- activemq::commands::SubscriptionInfo, 3098
- activemq::commands::TransactionId, 3218
- activemq::commands::TransactionInfo, 3241
- activemq::commands::WireFormatInfo, 3372
- activemq::commands::XATransactionId, 3411
- close
 - activemq::cmsutil::CachedConsumer, 954
 - activemq::cmsutil::CachedProducer, 958
 - activemq::cmsutil::PooledSession, 2507
 - activemq::commands::ActiveMQTempDestination, 500
 - activemq::commands::ConnectionControl, 1139
 - activemq::commands::ConsumerControl, 1254
 - activemq::core::ActiveMQConnection, 236
 - activemq::core::ActiveMQConsumer, 266
 - activemq::core::ActiveMQProducer, 408
 - activemq::core::ActiveMQSession, 447
 - activemq::core::ActiveMQSessionExecutor, 461
 - activemq::core::MessageDispatchChannel, 2225
 - activemq::transport::correlator::ResponseCorrelator, 2788
 - activemq::transport::failover::FailoverTransport, 1615
 - activemq::transport::inactivity::InactivityMonitor, 1734
 - activemq::transport::IOTransport, 1825
 - activemq::transport::mock::MockTransport, 2373
 - activemq::transport::tcp::TcpTransport, 3161
 - activemq::transport::TransportFilter, 3283
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2463
- cms::Closeable, 1021
- cms::Connection, 1132
- cms::Session, 2842
- decaf::internal::io::StandardErrorOutputStream, 2995
- decaf::internal::io::StandardInputStream, 3001
- decaf::internal::io::StandardOutputStream, 3009
- decaf::io::BlockingByteArrayInputStream, 726

- decaf::io::BufferedInputStream, 810
- decaf::io::BufferedOutputStream, 815
- decaf::io::ByteArrayInputStream, 894
- decaf::io::ByteArrayOutputStream, 902
- decaf::io::Closeable, 1019
- decaf::io::FilterInputStream, 1633
- decaf::io::FilterOutputStream, 1642
- decaf::net::BufferedSocket, 818
- decaf::net::ServerSocket, 2838
- decaf::net::SocketInputStream, 2979
- decaf::net::SocketOutputStream, 2985
- decaf::net::TcpSocket, 3154
- decaf::util::logging::StreamHandler, 3078
- closed
 - decaf::io::FilterInputStream, 1639
 - decaf::io::FilterOutputStream, 1646
- CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 1022
- cluster
 - activemq::commands::Message, 2161
- cms, 112
- cms/Config.h
 - CMS_API, 3590
- cms::BytesMessage, 938
 - ~BytesMessage, 941
 - clone, 941
 - getBodyBytes, 941
 - getBodyLength, 942
 - readBoolean, 942
 - readByte, 942
 - readBytes, 943
 - readChar, 944
 - readDouble, 944
 - readFloat, 944
 - readInt, 945
 - readLong, 945
 - readShort, 945
 - readString, 946
 - readUnsignedShort, 946
 - readUTF, 947
 - reset, 947
 - setBodyBytes, 947
 - writeBoolean, 947
 - writeByte, 948
 - writeBytes, 948, 949
 - writeChar, 949
 - writeDouble, 949
 - writeFloat, 950
 - writeInt, 950
 - writeLong, 950
 - writeShort, 951
 - writeString, 951
 - writeUnsignedShort, 951
 - writeUTF, 952
- cms::Closeable, 1021
 - ~Closeable, 1021
 - close, 1021
- cms::CMSException, 1031
 - ~CMSException, 1032
 - CMSException, 1032
 - getCause, 1032
 - getMessage, 1032
 - getStackTrace, 1032
 - getStackTraceString, 1033
 - printStackTrace, 1033
 - setMark, 1033
 - what, 1033
- cms::CMSProperties, 1036
 - ~CMSProperties, 1037
 - clear, 1037
 - clone, 1037
 - copy, 1037
 - getProperty, 1037
 - hasProperty, 1038
 - isEmpty, 1038
 - remove, 1038
 - setProperty, 1038
 - toArray, 1039
 - toString, 1039
- cms::CMSSecurityException, 1040
 - ~CMSSecurityException, 1040
 - CMSSecurityException, 1040
- cms::Connection, 1131
 - ~Connection, 1132
 - close, 1132
 - createSession, 1132, 1133
 - getClientID, 1133
 - getExceptionListener, 1133
 - getMetaData, 1133
 - setExceptionListener, 1134
- cms::ConnectionFactory, 1184
 - ~ConnectionFactory, 1185
 - createCMSConnectionFactory, 1185
 - createConnection, 1185, 1186
- cms::ConnectionMetaData, 1237
 - ~ConnectionMetaData, 1238
 - getCMSMajorVersion, 1238
 - getCMSMinorVersion, 1238
 - getCMSProviderName, 1238
 - getCMSVersion, 1238
 - getCMSXPropertyNames, 1239
 - getProviderMajorVersion, 1239
 - getProviderMinorVersion, 1239
 - getProviderVersion, 1240
- cms::DeliveryMode, 1478
 - ~DeliveryMode, 1479
 - DELIVERY_MODE, 1478

- NON_PERSISTENT, 1478
- PERSISTENT, 1478
- cms::Destination, 1480
 - ~Destination, 1481
 - clone, 1481
 - copy, 1481
 - DestinationType, 1480
 - getCMSProperties, 1481
 - getDestinationType, 1481
 - QUEUE, 1480
 - TEMPORARY_QUEUE, 1480
 - TEMPORARY_TOPIC, 1480
 - TOPIC, 1480
- cms::ExceptionListener, 1581
 - ~ExceptionListener, 1581
 - onException, 1581
- cms::IllegalStateException, 1729
 - ~IllegalStateException, 1729
 - IllegalStateException, 1729
- cms::InvalidClientIdException, 1808
 - ~InvalidClientIdException, 1808
 - InvalidClientIdException, 1808
- cms::InvalidDestinationException, 1809
 - ~InvalidDestinationException, 1809
 - InvalidDestinationException, 1809
- cms::InvalidSelectorException, 1816
 - ~InvalidSelectorException, 1816
 - InvalidSelectorException, 1816
- cms::MapMessage, 2106
 - ~MapMessage, 2108
 - getBoolean, 2108
 - getByte, 2108
 - getBytes, 2109
 - getChar, 2109
 - getDouble, 2109
 - getFloat, 2110
 - getInt, 2110
 - getLong, 2110
 - getMapNames, 2111
 - getShort, 2111
 - getString, 2111
 - itemExists, 2112
 - setBoolean, 2112
 - setByte, 2112
 - setBytes, 2113
 - setChar, 2113
 - setDouble, 2113
 - setFloat, 2114
 - setInt, 2114
 - setLong, 2114
 - setShort, 2115
 - setString, 2115
- cms::Message, 2163
 - ~Message, 2167
 - acknowledge, 2167
 - clearBody, 2167
 - clearProperties, 2168
 - clone, 2168
 - getBooleanProperty, 2169
 - getByteProperty, 2169
 - getCMSCorrelationID, 2169
 - getCMSDeliveryMode, 2170
 - getCMSDestination, 2170
 - getCMSExpiration, 2171
 - getCMSMessageID, 2171
 - getCMSPriority, 2172
 - getCMSRedelivered, 2172
 - getCMSReplyTo, 2173
 - getCMSTimestamp, 2173
 - getCMSType, 2174
 - getDoubleProperty, 2174
 - getFloatProperty, 2175
 - getIntProperty, 2175
 - getLongProperty, 2176
 - getPropertyNames, 2176
 - getShortProperty, 2177
 - getStringProperty, 2177
 - propertyExists, 2178
 - setBooleanProperty, 2178
 - setByteProperty, 2179
 - setCMSCorrelationID, 2179
 - setCMSDeliveryMode, 2180
 - setCMSDestination, 2180
 - setCMSExpiration, 2181
 - setCMSMessageID, 2181
 - setCMSPriority, 2181
 - setCMSRedelivered, 2182
 - setCMSReplyTo, 2182
 - setCMSTimestamp, 2183
 - setCMSType, 2183
 - setDoubleProperty, 2184
 - setFloatProperty, 2184
 - setIntProperty, 2185
 - setLongProperty, 2185
 - setShortProperty, 2186
 - setStringProperty, 2186
- cms::MessageConsumer, 2214
 - ~MessageConsumer, 2215
 - getMessageListener, 2215
 - getMessageSelector, 2215
 - receive, 2215, 2216
 - receiveNoWait, 2216
 - setMessageListener, 2216
- cms::MessageEOFException, 2277
 - ~MessageEOFException, 2277
 - MessageEOFException, 2277
- cms::MessageFormatException, 2278
 - ~MessageFormatException, 2278

- MessageFormatException, 2278
- cms::MessageListener, 2304
 - ~MessageListener, 2304
 - onMessage, 2304
- cms::MessageNotReadableException, 2330
 - ~MessageNotReadableException, 2330
 - MessageNotReadableException, 2330
- cms::MessageNotWriteableException, 2331
 - ~MessageNotWriteableException, 2331
 - MessageNotWriteableException, 2331
- cms::MessageProducer, 2332
 - ~MessageProducer, 2333
 - getDeliveryMode, 2333
 - getDisableMessageID, 2334
 - getDisableMessageTimeStamp, 2334
 - getPriority, 2334
 - getTimeToLive, 2334
 - send, 2335, 2336
 - setDeliveryMode, 2337
 - setDisableMessageID, 2337
 - setDisableMessageTimeStamp, 2337
 - setPriority, 2338
 - setTimeToLive, 2338
- cms::ObjectMessage, 2438
 - ~ObjectMessage, 2438
- cms::Queue, 2674
 - ~Queue, 2674
 - getQueueName, 2674
- cms::QueueBrowser, 2675
 - ~QueueBrowser, 2675
 - getEnumeration, 2675
 - getMessageSelector, 2675
 - getQueue, 2676
- cms::Session, 2839
 - ~Session, 2842
 - AcknowledgeMode, 2842
 - AUTO_ACKNOWLEDGE, 2842
 - CLIENT_ACKNOWLEDGE, 2842
 - close, 2842
 - commit, 2843
 - createBrowser, 2843
 - createBytesMessage, 2844
 - createConsumer, 2844, 2845
 - createDurableConsumer, 2846
 - createMapMessage, 2846
 - createMessage, 2846
 - createProducer, 2847
 - createQueue, 2847
 - createStreamMessage, 2847
 - createTemporaryQueue, 2848
 - createTemporaryTopic, 2848
 - createTextMessage, 2848, 2849
 - createTopic, 2849
 - DUPS_OK_ACKNOWLEDGE, 2842
 - getAcknowledgeMode, 2849
 - INDIVIDUAL_ACKNOWLEDGE, 2842
 - isTransacted, 2850
 - recover, 2850
 - rollback, 2850
 - SESSION_TRANSACTED, 2842
 - unsubscribe, 2851
- cms::Startable, 3014
 - ~Startable, 3014
 - start, 3014
- cms::Stoppable, 3076
 - ~Stoppable, 3076
 - stop, 3076
- cms::StreamMessage, 3081
 - ~StreamMessage, 3084
 - readBoolean, 3084
 - readByte, 3084
 - readBytes, 3084, 3085
 - readChar, 3086
 - readDouble, 3086
 - readFloat, 3086
 - readInt, 3087
 - readLong, 3087
 - readShort, 3088
 - readString, 3088
 - readUnsignedShort, 3088
 - writeBoolean, 3089
 - writeByte, 3089
 - writeBytes, 3090
 - writeChar, 3090
 - writeDouble, 3091
 - writeFloat, 3091
 - writeInt, 3091
 - writeLong, 3092
 - writeShort, 3092
 - writeString, 3092
 - writeUnsignedShort, 3093
- cms::TemporaryQueue, 3166
 - ~TemporaryQueue, 3166
 - destroy, 3166
 - getQueueName, 3166
- cms::TemporaryTopic, 3168
 - ~TemporaryTopic, 3168
 - destroy, 3168
 - getTopicName, 3168
- cms::TextMessage, 3170
 - ~TextMessage, 3170
 - getText, 3170
 - setText, 3171
- cms::Topic, 3215
 - ~Topic, 3215
 - getTopicName, 3215
- cms::UnsupportedOperationException, 3306
 - ~UnsupportedOperationException, 3306

- UnsupportedOperationException, 3306
- CMS_API
 - cms/Config.h, 3590
- CmsAccessor
 - activemq::cmsutil::CmsAccessor, 1025
- CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 1029
- CMSException
 - cms::CMSException, 1032
- CMSExceptionSupport.h
 - AMQ_CATCH_ALL_THROW_-CMSEXCEPTION, 3586
- CMSSecurityException
 - cms::CMSSecurityException, 1040
- CmsTemplate
 - activemq::cmsutil::CmsTemplate, 1044
- command
 - activemq::commands::ControlCommand, 1333
- commandId
 - activemq::commands::PartialCommand, 2476
- COMMIT
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- commit
 - activemq::cmsutil::PooledSession, 2507
 - activemq::core::ActiveMQConsumer, 266
 - activemq::core::ActiveMQSession, 447
 - activemq::core::ActiveMQTransactionContext, 623
 - cms::Session, 2843
- compact
 - decaf::internal::nio::ByteBuffer, 879
 - decaf::internal::nio::CharArrayBuffer, 994
 - decaf::internal::nio::DoubleArrayBuffer, 1551
 - decaf::internal::nio::FloatArrayBuffer, 1661
 - decaf::internal::nio::IntArrayBuffer, 1747
 - decaf::internal::nio::LongArrayBuffer, 2074
 - decaf::internal::nio::ShortArrayBuffer, 2918
 - decaf::nio::ByteBuffer, 921
 - decaf::nio::CharBuffer, 1004
 - decaf::nio::DoubleBuffer, 1559
 - decaf::nio::FloatBuffer, 1668
 - decaf::nio::IntBuffer, 1754
 - decaf::nio::LongBuffer, 2082
 - decaf::nio::ShortBuffer, 2926
- COMPARATOR
 - activemq::commands::BrokerId, 752
 - activemq::commands::ConnectionId, 1188
 - activemq::commands::ConsumerId, 1276
 - activemq::commands::LocalTransactionId, 1994
 - activemq::commands::MessageId, 2280
 - activemq::commands::ProducerId, 2607
 - activemq::commands::SessionId, 2854
 - activemq::commands::TransactionId, 3217
 - activemq::commands::XATransactionId, 3411
- comparator
 - decaf::util::PriorityQueue, 2574
- compare
 - decaf::lang::Double, 1539
 - decaf::lang::Float, 1649
 - decaf::lang::PointerComparator, 2504
 - decaf::util::Comparator, 1086
 - decaf::util::comparators::Less, 1981
- compareAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 631
 - decaf::util::concurrent::atomic::AtomicInteger, 635
 - decaf::util::concurrent::atomic::AtomicReference, 642
- compareTo
 - activemq::commands::BrokerId, 752
 - activemq::commands::ConnectionId, 1188
 - activemq::commands::ConsumerId, 1276
 - activemq::commands::LocalTransactionId, 1994
 - activemq::commands::MessageId, 2280
 - activemq::commands::ProducerId, 2607
 - activemq::commands::SessionId, 2854
 - activemq::commands::TransactionId, 3218
 - activemq::commands::XATransactionId, 3411
 - decaf::lang::Boolean, 733
 - decaf::lang::Byte, 842
 - decaf::lang::Character, 984
 - decaf::lang::Comparable, 1083
 - decaf::lang::Double, 1540
 - decaf::lang::Float, 1650
 - decaf::lang::Integer, 1765
 - decaf::lang::Long, 2060
 - decaf::lang::Short, 2908
 - decaf::net::URI, 3312
 - decaf::nio::ByteBuffer, 922
 - decaf::nio::CharBuffer, 1004
 - decaf::nio::DoubleBuffer, 1560
 - decaf::nio::FloatBuffer, 1669
 - decaf::nio::IntBuffer, 1755
 - decaf::nio::LongBuffer, 2083
 - decaf::nio::ShortBuffer, 2927
 - decaf::util::concurrent::TimeUnit, 3207
 - decaf::util::Date, 1469

- decaf::util::UUID, 3358
- COMPOSITE_SEPARATOR
 - activemq::commands::ActiveMQDestination, 283
- CompositeData
 - activemq::util::CompositeData, 1089
- CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 1093
- compressed
 - activemq::commands::Message, 2161
- Concurrent.h
 - synchronized, 4126
 - WAIT_INFINITE, 4126
- ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 1106
- condition
 - decaf::util::concurrent::ConditionHandle, 1124
- ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 1124
- configure
 - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2123
 - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2121
 - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2124
 - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2122
 - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2125
- CONNECT
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- connect
 - decaf::net::BufferedSocket, 818
 - decaf::net::Socket, 2965
 - decaf::net::TcpSocket, 3154
- CONNECTED
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- ConnectException
 - decaf::net::ConnectException, 1128, 1129
- connection
 - activemq::commands::ActiveMQTempDestination, 502
- CONNECTION_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 283
- CONNECTION_ALWAYS_SYNC_SEND
 - activemq::core::ActiveMQConstants, 262
- CONNECTION_CLOSETIMEOUT
 - activemq::core::ActiveMQConstants, 262
- CONNECTION_PRODUCER_WINDOW_SIZE
 - activemq::core::ActiveMQConstants, 262
- CONNECTION_SENDTIMEOUT
 - activemq::core::ActiveMQConstants, 262
- CONNECTION_USE_ASYNC_SEND
 - activemq::core::ActiveMQConstants, 262
- ConnectionControl
 - activemq::commands::ConnectionControl, 1136
- ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1145
 - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1157
 - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1141
 - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1149
 - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1153
- ConnectionError
 - activemq::commands::ConnectionError, 1161
- ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1169
 - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1181
 - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1165
 - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1173
 - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1177
- ConnectionId
 - activemq::commands::ConnectionId, 1188
- connectionId
 - activemq::commands::BrokerInfo, 782
 - activemq::commands::ConnectionError, 1163
 - activemq::commands::ConnectionInfo, 1216
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::DestinationInfo, 1488
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::ProducerId, 2609
 - activemq::commands::RemoveSubscriptionInfo, 2731
 - activemq::commands::SessionId, 2856

- activemq::commands::TransactionInfo, 3243
- ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1196
 - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1208
 - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1192
 - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1200
 - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1204
- ConnectionInfo
 - activemq::commands::ConnectionInfo, 1212
- ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1226
 - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1234
 - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1218
 - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1222
 - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1230
- ConnectionState
 - activemq::state::ConnectionState, 1242
- ConnectionStateTracker
 - activemq::state::ConnectionStateTracker, 1246
- CONSUMER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 283
- CONSUMER_DISPATCHASYNC
 - activemq::core::ActiveMQConstants, 261
- CONSUMER_EXCLUSIVE
 - activemq::core::ActiveMQConstants, 261
- CONSUMER_NOLOCAL
 - activemq::core::ActiveMQConstants, 261
- CONSUMER_PREFETCHSIZE
 - activemq::core::ActiveMQConstants, 261
- CONSUMER_PRIORITY
 - activemq::core::ActiveMQConstants, 261
- CONSUMER_RETROACTIVE
 - activemq::core::ActiveMQConstants, 261
- CONSUMER_SELECTOR
 - activemq::core::ActiveMQConstants, 261
- ConsumerControl
 - activemq::commands::ConsumerControl, 1251
- ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1264
 - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1272
 - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1256
 - activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1260
 - activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1268
- activemq::commands::ConsumerId, 1276
- consumerId
 - activemq::commands::ConsumerControl, 1254
 - activemq::commands::ConsumerInfo, 1307
 - activemq::commands::MessageAck, 2193
 - activemq::commands::MessageDispatch, 2223
 - activemq::commands::MessageDispatchNotification, 2256
 - activemq::commands::MessagePull, 2350
- ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1289
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1297
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1281
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1285
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1293
- ConsumerInfo
 - activemq::commands::ConsumerInfo, 1302
- ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1314
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1326
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1310
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1318
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1322
- ConsumerState
 - activemq::state::ConsumerState, 1329
- contains
 - decaf::util::AbstractCollection, 152
 - decaf::util::Collection, 1058
 - decaf::util::concurrent::SynchronousQueue, 3140
 - decaf::util::StiList, 3024

- decaf::util::StlSet, 3054
- containsAll
 - decaf::util::AbstractCollection, 152
 - decaf::util::Collection, 1058
 - decaf::util::concurrent::SynchronousQueue, 3140
- containsKey
 - decaf::util::concurrent::ConcurrentStlMap, 1107
 - decaf::util::Map, 2096
 - decaf::util::StlMap, 3034
- containsValue
 - decaf::util::concurrent::ConcurrentStlMap, 1107
 - decaf::util::Map, 2097
 - decaf::util::StlMap, 3035
- content
 - activemq::commands::Message, 2161
- ControlCommand
 - activemq::commands::ControlCommand, 1331
- ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1343
 - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1351
 - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1335
 - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1339
 - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1347
- convert
 - activemq::util::PrimitiveValueConverter, 2553
 - decaf::util::concurrent::TimeUnit, 3208
- convertConsumerId
 - activemq::wireformat::stomp::StompHelper, 3067
- convertDestination
 - activemq::wireformat::stomp::StompHelper, 3067, 3068
- convertMessageId
 - activemq::wireformat::stomp::StompHelper, 3068
- convertProducerId
 - activemq::wireformat::stomp::StompHelper, 3068, 3069
- convertProperties
 - activemq::wireformat::stomp::StompHelper, 3069
- convertToCMSException
 - activemq::exceptions::ActiveMQException, 306
- convertTransactionId
 - activemq::wireformat::stomp::StompHelper, 3069, 3070
- copy
 - activemq::commands::ActiveMQQueue, 420
 - activemq::commands::ActiveMQTempQueue, 524
 - activemq::commands::ActiveMQTempTopic, 549
 - activemq::commands::ActiveMQTopic, 599
 - activemq::util::ActiveMQProperties, 415
 - activemq::wireformat::stomp::StompFrame, 3062
 - cms::CMSProperties, 1037
 - cms::Destination, 1481
 - decaf::util::AbstractCollection, 153
 - decaf::util::concurrent::ConcurrentStlMap, 1107, 1108
 - decaf::util::Map, 2097
 - decaf::util::Properties, 2659
 - decaf::util::StlList, 3024
 - decaf::util::StlMap, 3035
 - decaf::util::StlSet, 3055
 - copyDataStructure
 - activemq::commands::ActiveMQBlobMessage, 173
 - activemq::commands::ActiveMQBytesMessage, 201
 - activemq::commands::ActiveMQDestination, 277
 - activemq::commands::ActiveMQMapMessage, 312
 - activemq::commands::ActiveMQMessage, 343
 - activemq::commands::ActiveMQObjectMessage, 384
 - activemq::commands::ActiveMQQueue, 420
 - activemq::commands::ActiveMQStreamMessage, 467
 - activemq::commands::ActiveMQTempDestination, 500
 - activemq::commands::ActiveMQTempQueue, 524
 - activemq::commands::ActiveMQTempTopic, 549
 - activemq::commands::ActiveMQTextMessage, 575
 - activemq::commands::ActiveMQTopic, 599
 - activemq::commands::BaseCommand, 651
 - activemq::commands::BaseDataStructure, 716

- activemq::commands::BooleanExpression, 737
- activemq::commands::BrokerError, 746
- activemq::commands::BrokerId, 752
- activemq::commands::BrokerInfo, 777
- activemq::commands::ConnectionControl, 1136
- activemq::commands::ConnectionError, 1161
- activemq::commands::ConnectionId, 1188
- activemq::commands::ConnectionInfo, 1212
- activemq::commands::ConsumerControl, 1251
- activemq::commands::ConsumerId, 1276
- activemq::commands::ConsumerInfo, 1302
- activemq::commands::ControlCommand, 1331
- activemq::commands::DataArrayResponse, 1357
- activemq::commands::DataResponse, 1396
- activemq::commands::DataStructure, 1462
- activemq::commands::DestinationInfo, 1485
- activemq::commands::DiscoveryEvent, 1512
- activemq::commands::ExceptionResponse, 1583
- activemq::commands::FlushCommand, 1677
- activemq::commands::IntegerResponse, 1778
- activemq::commands::JournalQueueAck, 1835
- activemq::commands::JournalTopicAck, 1859
- activemq::commands::JournalTrace, 1884
- activemq::commands::JournalTransaction, 1907
- activemq::commands::KeepAliveInfo, 1931
- activemq::commands::LastPartialCommand, 1959
- activemq::commands::LocalTransactionId, 1994
- activemq::commands::Message, 2150
- activemq::commands::MessageAck, 2189
- activemq::commands::MessageDispatch, 2220
- activemq::commands::MessageDispatchNotification, 2252
- activemq::commands::MessageId, 2280
- activemq::commands::MessagePull, 2347
- activemq::commands::NetworkBridgeFilter, 2394
- activemq::commands::PartialCommand, 2474
- activemq::commands::ProducerAck, 2580
- activemq::commands::ProducerId, 2607
- activemq::commands::ProducerInfo, 2632
- activemq::commands::RemoveInfo, 2704
- activemq::commands::RemoveSubscriptionInfo, 2728
- activemq::commands::ReplayCommand, 2753
- activemq::commands::Response, 2782
- activemq::commands::SessionId, 2854
- activemq::commands::SessionInfo, 2878
- activemq::commands::ShutdownInfo, 2935
- activemq::commands::SubscriptionInfo, 3098
- activemq::commands::TransactionId, 3218
- activemq::commands::TransactionInfo, 3241
- activemq::commands::WireFormatInfo, 3372
- activemq::commands::XATransactionId, 3411
- correlationId
 - activemq::commands::Message, 2161
 - activemq::commands::MessagePull, 2350
 - activemq::commands::Response, 2784
- countDown
 - decaf::util::concurrent::CountDownLatch, 1355
- CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1354
- CounterType
 - decaf::lang::Pointer, 2499
- countTokens
 - decaf::util::StringTokenizer, 3095
- create
 - activemq::transport::failover::FailoverTransportFactory, 1625
 - activemq::transport::mock::MockTransportFactory, 2383
 - activemq::transport::tcp::TcpTransportFactory, 3164
 - activemq::transport::TransportFactory, 3279
 - activemq::util::CMSExceptionSupport, 1034
 - decaf::internal::util::concurrent::ConditionImpl, 1126
 - decaf::internal::util::concurrent::MutexImpl, 2391
 - decaf::net::URI, 3312
- createBrowser

- activemq::cmsutil::PooledSession, 2508
- activemq::core::ActiveMQSession, 447, 448
- cms::Session, 2843
- createByteBuffer
 - decaf::internal::nio::BufferFactory, 826
- createBytesMessage
 - activemq::cmsutil::PooledSession, 2508, 2509
 - activemq::core::ActiveMQSession, 448
 - cms::Session, 2844
- createCachedConsumer
 - activemq::cmsutil::PooledSession, 2509
- createCachedProducer
 - activemq::cmsutil::PooledSession, 2509
- createCharBuffer
 - decaf::internal::nio::BufferFactory, 827
- createCMSConnectionFactory
 - cms::ConnectionFactory, 1185
- createComposite
 - activemq::transport::failover::FailoverTransportFactory, 1625
 - activemq::transport::mock::MockTransportFactory, 2383
 - activemq::transport::tcp::TcpTransportFactory, 3164
 - activemq::transport::TransportFactory, 3280
- createConnection
 - activemq::cmsutil::CmsAccessor, 1025
 - activemq::core::ActiveMQConnectionFactory, 245, 246
 - cms::ConnectionFactory, 1185, 1186
- createConsumer
 - activemq::cmsutil::PooledSession, 2510, 2511
 - activemq::core::ActiveMQSession, 449
 - cms::Session, 2844, 2845
- createDestination
 - activemq::commands::ActiveMQDestination, 277
- createDoubleBuffer
 - decaf::internal::nio::BufferFactory, 828
- createDurableConsumer
 - activemq::cmsutil::PooledSession, 2511
 - activemq::core::ActiveMQSession, 450
 - cms::Session, 2846
- createFloatBuffer
 - decaf::internal::nio::BufferFactory, 829
- createIntBuffer
 - decaf::internal::nio::BufferFactory, 830
- createLongBuffer
 - decaf::internal::nio::BufferFactory, 831
- createMapMessage
 - activemq::cmsutil::PooledSession, 2512
- activemq::core::ActiveMQSession, 450
- cms::Session, 2846
- createMessage
 - activemq::cmsutil::MessageCreator, 2218
 - activemq::cmsutil::PooledSession, 2512
 - activemq::core::ActiveMQSession, 450
 - cms::Session, 2846
- createMessageEOFException
 - activemq::util::CMSExceptionSupport, 1034
- createMessageFormatException
 - activemq::util::CMSExceptionSupport, 1034
- createNegotiator
 - activemq::wireformat::openwire::OpenWireFormat, 2451
 - activemq::wireformat::stomp::StompWireFormat, 3072
 - activemq::wireformat::WireFormat, 3364
- createObject
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1419
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller, 182
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller, 218
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller, 327
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 350
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller, 391
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 428
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller, 484
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempObjectMarshaller, 533
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempObjectMarshaller, 558
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller, 583
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 607
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 760
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 788
 - activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller, 1145
 - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1169
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1196

activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2624
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2624
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	2649
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2708
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	2741
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	2773
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2807
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2862
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	2882
activemq::wireformat::openwire::marshal::v1::DiscoveryInfoMarshaller	2938
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	3103
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3246
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	3396
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3420
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	194
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	230
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	339
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	362
activemq::wireformat::openwire::marshal::v1::LastBatchCommandMarshaller	403
activemq::wireformat::openwire::marshal::v1::LocalTransactionMarshaller	440
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	496
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	545
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	570
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	595
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	619
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	772
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	800
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	1157
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMarshaller	
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	
activemq::wireformat::openwire::marshal::v2::ConnectionContextMarshaller	

activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2478
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	1181
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	1208
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	1234
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	1272
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	1297
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	1326
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	1351
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	1376
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	1415
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	1490
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	1516
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	1586
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	1680
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	1781
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	1839
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	1864
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	1887
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller	1911
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	1934
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller	1962
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1998
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller	2195
activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller	2232
activemq::wireformat::openwire::marshal::v3::ActiveMQTemp5Marshaller	2258
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMarshaller	2285
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2352
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2398

activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2356	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
784		
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2402	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFactory
1141		
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2482	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1165		
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2588	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1192		
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2616	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1218		
activemq::wireformat::openwire::marshal::v3::ConsumerGroupViewMarshaller	2653	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1256		
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2716	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1281		
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	2745	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1310		
activemq::wireformat::openwire::marshal::v3::ContainerCommandMarshaller	2761	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1335		
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	2797	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1360		
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	2870	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1399		
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	2898	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1494		
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	2946	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1520		
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	3111	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1594		
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	3254	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
1688		
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller	3380	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
1789		
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	3416	activemq::wireformat::openwire::marshal::v3::XATransactionInfoMarshaller
1847		
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	186	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller
1872		
activemq::wireformat::openwire::marshal::v3::JournalTranqMarshaller	222	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller
1895		
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	331	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller
1915		
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	354	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
1942		
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	395	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller
1970		
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	432	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
2002		
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	488	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller
2203		
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	537	activemq::wireformat::openwire::marshal::v4::ActiveMQTempMarshaller
2240		
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	562	activemq::wireformat::openwire::marshal::v4::ActiveMQTempMarshaller
2266		
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	587	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMarshaller
2293		

activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2262
611	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2289
764	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2368
792	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2414
1149	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2486
1173	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	2592
1200	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	2628
1222	
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller	2645
1260	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	2720
1285	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	2749
1318	
activemq::wireformat::openwire::marshal::v4::ContactGroupMarshaller	2769
1339	
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	2812
1364	
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	2874
1411	
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	2886
1498	
activemq::wireformat::openwire::marshal::v4::DiscoveryInfoMarshaller	2942
1524	
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	3107
1598	
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	3250
1692	
activemq::wireformat::openwire::marshal::v4::IntegrationResponseMarshaller	3392
1793	
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	3424
1843	
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	190
1876	
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	226
1891	
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	335
1919	
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	358
1946	
activemq::wireformat::openwire::marshal::v4::LastReceivedCommandMarshaller	399
1974	
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller	436
2006	
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	492
2207	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	541
2236	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
	2262
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2289
	2368
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2414
	2486
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2592
	2628
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2645
	2720
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2749
	2769
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2812
	2874
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2886
	2942
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	3107
	3250
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	3392
	3424
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	190
	226
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	335
	358
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	399
	436
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	492
	541

activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, openwire::marshal::v5::MessageAckMarshaller, 566
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, openwire::marshal::v5::MessageDispatcher, 591
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, openwire::marshal::v5::MessageDispatcher, 615
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, openwire::marshal::v5::MessageIdMarshaller, 768
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, openwire::marshal::v5::MessagePullMarshaller, 796
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, openwire::marshal::v5::NetworkBridgeFactory, 1153
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, openwire::marshal::v5::PartialCommand, 1177
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, openwire::marshal::v5::ProducerAckMarshaller, 1204
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, openwire::marshal::v5::ProducerIdMarshaller, 1230
 activemq::wireformat::openwire::marshal::v5::ConsumerGroupWithMarshaller, openwire::marshal::v5::ProducerInfoMarshaller, 1268
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, openwire::marshal::v5::RemoveInfoMarshaller, 1293
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, openwire::marshal::v5::RemoveSubscription, 1322
 activemq::wireformat::openwire::marshal::v5::ContactCommandMarshaller, openwire::marshal::v5::ReplayCommand, 1347
 activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller, openwire::marshal::v5::ResponseMarshaller, 1372
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, openwire::marshal::v5::SessionIdMarshaller, 1407
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, openwire::marshal::v5::SessionInfoMarshaller, 1506
 activemq::wireformat::openwire::marshal::v5::DiscardResponseMarshaller, openwire::marshal::v5::ShutdownInfoMarshaller, 1528
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, openwire::marshal::v5::SubscriptionInfoMarshaller, 1602
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, openwire::marshal::v5::TransactionInfoMarshaller, 1696
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, openwire::marshal::v5::WireFormatInfoMarshaller, 1797
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, openwire::marshal::v5::XATransactionInfoMarshaller, 1855
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, activemq::cmsutil::PooledSession, 2512
 activemq::wireformat::openwire::marshal::v5::JournalQueueMarshaller, activemq::MQSession, 451
 activemq::wireformat::openwire::marshal::v5::JournalQueueStringMarshaller, cms::Session, 2847
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, activemq::util::URISupport, 3332
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, activemq::cmsutil::PooledSession, 2513
 activemq::wireformat::openwire::marshal::v5::LastReceivedCommandMarshaller, activemq::MQSession, 451
 activemq::wireformat::openwire::marshal::v5::LastReceivedCommandMarshaller, cms::Session, 2847
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, activemq::cmsutil::CmsAccessor, 1025
 2014

- activemq::core::ActiveMQConnection, 236, 237
- cms::Connection, 1132, 1133
- createShortBuffer
 - decaf::internal::nio::BufferFactory, 832
- createSocket
 - decaf::net::SocketFactory, 2975
- createStreamMessage
 - activemq::cmsutil::PooledSession, 2513
 - activemq::core::ActiveMQSession, 451
 - cms::Session, 2847
- createTemporaryName
 - activemq::commands::ActiveMQDestination, 277
- createTemporaryQueue
 - activemq::cmsutil::PooledSession, 2513
 - activemq::core::ActiveMQSession, 452
 - cms::Session, 2848
- createTemporaryTopic
 - activemq::cmsutil::PooledSession, 2514
 - activemq::core::ActiveMQSession, 452
 - cms::Session, 2848
- createTextMessage
 - activemq::cmsutil::PooledSession, 2514
 - activemq::core::ActiveMQSession, 452
 - cms::Session, 2848, 2849
- createTopic
 - activemq::cmsutil::PooledSession, 2515
 - activemq::core::ActiveMQSession, 453
 - cms::Session, 2849
- createWireFormat
 - activemq::transport::AbstractTransportFactory, 169
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2460
 - activemq::wireformat::stomp::StompWireFormatFactory, 3075
 - activemq::wireformat::WireFormatFactory, 3367
- createX500Principal
 - decaf::security_provider::SecurityProvider, 2822
- criticalSection
 - decaf::util::concurrent::ConditionHandle, 1124
- CUNSUMER_MAXPENDINGMSGLIMIT
 - activemq::core::ActiveMQConstants, 261
- currentTimeMillis
 - decaf::lang::System, 3146
- data
 - activemq::commands::DataArrayResponse, 1358
 - activemq::commands::DataResponse, 1397
 - activemq::commands::PartialCommand, 2476
 - DataArrayResponse
 - activemq::commands::DataArrayResponse, 1357
 - DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataArrayResponse, 1368
 - activemq::wireformat::openwire::marshal::v2::DataArrayResponse, 1376
 - activemq::wireformat::openwire::marshal::v3::DataArrayResponse, 1360
 - activemq::wireformat::openwire::marshal::v4::DataArrayResponse, 1364
 - activemq::wireformat::openwire::marshal::v5::DataArrayResponse, 1372
 - DataInputStream
 - decaf::io::DataInputStream, 1380
 - DataOutputStream
 - decaf::io::DataOutputStream, 1389
 - DataResponse
 - activemq::commands::DataResponse, 1396
 - DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataResponseMessage, 1403
 - activemq::wireformat::openwire::marshal::v2::DataResponseMessage, 1415
 - activemq::wireformat::openwire::marshal::v3::DataResponseMessage, 1399
 - activemq::wireformat::openwire::marshal::v4::DataResponseMessage, 1411
 - activemq::wireformat::openwire::marshal::v5::DataResponseMessage, 1407
 - DataStructure
 - activemq::commands::Message, 2161
 - Date
 - decaf::util::Date, 1468
 - DAYS
 - decaf::util::concurrent::TimeUnit, 3213
 - Debug
 - decaf::util::logging, 144
 - debug
 - decaf::util::logging::Logger, 2030
 - decaf::util::logging::SimpleLogger, 2963
 - decaf, 115
 - decaf/lang/exceptions/ExceptionDefines.h
 - DECAF_CATCH_EXCEPTION_-CONVERT, 3536
 - DECAF_CATCH_NOTHROW, 3536
 - DECAF_CATCH_RETHROW, 3537
 - DECAF_CATCHALL_NOTHROW, 3537
 - DECAF_CATCHALL_THROW, 3537
 - decaf/util/Config.h
 - DECAF_API, 3591

- DECAF_UNUSED, 3591
- decaf::internal, 116
- decaf::internal::AprPool, 628
 - ~AprPool, 628
 - AprPool, 628
 - cleanup, 628
 - getAprPool, 628
 - getGlobalPool, 628
- decaf::internal::DecafRuntime, 1472
 - ~DecafRuntime, 1472
 - DecafRuntime, 1472
 - getGlobalPool, 1472
- decaf::internal::io, 117
- decaf::internal::io::StandardErrorOutputStream, 2994
 - ~StandardErrorOutputStream, 2995
 - close, 2995
 - flush, 2995
 - lock, 2995
 - notify, 2996
 - notifyAll, 2996
 - StandardErrorOutputStream, 2995
 - tryLock, 2996
 - unlock, 2997
 - wait, 2997, 2998
 - write, 2998, 2999
- decaf::internal::io::StandardInputStream, 3000
 - ~StandardInputStream, 3001
 - available, 3001
 - close, 3001
 - lock, 3002
 - mark, 3002
 - markSupported, 3002
 - notify, 3002
 - notifyAll, 3003
 - read, 3003
 - reset, 3004
 - skip, 3004
 - StandardInputStream, 3001
 - tryLock, 3005
 - unlock, 3005
 - wait, 3005, 3006
- decaf::internal::io::StandardOutputStream, 3008
 - ~StandardOutputStream, 3009
 - close, 3009
 - flush, 3009
 - lock, 3009
 - notify, 3009
 - notifyAll, 3010
 - StandardOutputStream, 3009
 - tryLock, 3010
 - unlock, 3010
 - wait, 3011, 3012
 - write, 3012, 3013
- decaf::internal::net, 118
- decaf::internal::net::URLEncoderDecoder, 3319
 - ~URLEncoderDecoder, 3319
 - decode, 3319
 - encodeOthers, 3320
 - quoteIllegal, 3320
 - URLEncoderDecoder, 3319
 - validate, 3320
 - validateSimple, 3320
- decaf::internal::net::URIHelper, 3322
 - ~URIHelper, 3323
 - isValidDomainName, 3324
 - isValidHexChar, 3324
 - isValidHost, 3324
 - isValidIP4Word, 3324
 - isValidIP6Address, 3325
 - isValidIPv4Address, 3325
 - parseAuthority, 3325
 - parseURI, 3326
 - URIHelper, 3323
 - validateAuthority, 3326
 - validateFragment, 3326
 - validatePath, 3326
 - validateQuery, 3327
 - validateScheme, 3327
 - validateSsp, 3327
 - validateUserInfo, 3328
- decaf::internal::net::URITYPE, 3339
 - ~URITYPE, 3341
 - getAuthority, 3341
 - getFragment, 3341
 - getHost, 3341
 - getPath, 3341
 - getPort, 3341
 - getQuery, 3342
 - getScheme, 3342
 - getSchemeSpecificPart, 3342
 - getSource, 3342
 - getUserInfo, 3342
 - isAbsolute, 3342
 - isOpaque, 3343
 - isServerAuthority, 3343
 - isValid, 3343
 - setAbsolute, 3343
 - setAuthority, 3343
 - setFragment, 3343
 - setHost, 3344
 - setOpaque, 3344
 - setPath, 3344
 - setPort, 3344
 - setQuery, 3344
 - setScheme, 3344
 - setSchemeSpecificPart, 3345

- setServerAuthority, 3345
- setSource, 3345
- setUserInfo, 3345
- setValid, 3345
- URIType, 3341
- decaf::internal::nio, 119
- decaf::internal::nio::BufferFactory, 824
 - ~BufferFactory, 826
 - createByteBuffer, 826
 - createCharBuffer, 827
 - createDoubleBuffer, 828
 - createFloatBuffer, 829
 - createIntBuffer, 830
 - createLongBuffer, 831
 - createShortBuffer, 832
- decaf::internal::nio::ByteBuffer, 870
 - ~ByteBuffer, 875
 - array, 876
 - arrayOffset, 876
 - asCharBuffer, 876
 - asDoubleBuffer, 877
 - asFloatBuffer, 877
 - asIntBuffer, 877
 - asLongBuffer, 878
 - asReadOnlyBuffer, 878
 - asShortBuffer, 878
 - ByteBuffer, 874, 875
 - compact, 879
 - duplicate, 879
 - get, 879, 880
 - getChar, 880
 - getDouble, 881
 - getFloat, 881, 882
 - getInt, 882
 - getLong, 883
 - getShort, 883, 884
 - hasArray, 884
 - isReadOnly, 884
 - put, 885
 - putChar, 885, 886
 - putDouble, 886, 887
 - putFloat, 887, 888
 - putInt, 888
 - putLong, 889
 - putShort, 890
 - setReadOnly, 891
 - slice, 891
- decaf::internal::nio::ByteArrayPerspective, 908
 - ~ByteArrayPerspective, 911
 - ByteArrayPerspective, 909–911
 - getReferences, 911
 - returnRef, 911
 - takeRef, 912
- decaf::internal::nio::CharArrayBuffer, 990
 - ~CharArrayBuffer, 992
 - _array, 997
 - array, 993
 - arrayOffset, 993
 - asReadOnlyBuffer, 993
 - CharArrayBuffer, 991, 992
 - compact, 994
 - duplicate, 994
 - get, 994, 995
 - hasArray, 995
 - isReadOnly, 995
 - offset, 997
 - put, 995, 996
 - readOnly, 997
 - setReadOnly, 996
 - slice, 996
 - subSequence, 997
- decaf::internal::nio::DoubleArrayBuffer, 1548
 - ~DoubleArrayBuffer, 1550
 - array, 1550
 - arrayOffset, 1551
 - asReadOnlyBuffer, 1551
 - compact, 1551
 - DoubleArrayBuffer, 1549, 1550
 - duplicate, 1552
 - get, 1552
 - hasArray, 1553
 - isReadOnly, 1553
 - put, 1553, 1554
 - setReadOnly, 1554
 - slice, 1554
- decaf::internal::nio::FloatArrayBuffer, 1658
 - ~FloatArrayBuffer, 1660
 - array, 1660
 - arrayOffset, 1661
 - asReadOnlyBuffer, 1661
 - compact, 1661
 - duplicate, 1662
 - FloatArrayBuffer, 1659, 1660
 - get, 1662
 - hasArray, 1663
 - isReadOnly, 1663
 - put, 1663
 - setReadOnly, 1664
 - slice, 1664
- decaf::internal::nio::IntArrayBuffer, 1744
 - ~IntArrayBuffer, 1746
 - array, 1746
 - arrayOffset, 1747
 - asReadOnlyBuffer, 1747
 - compact, 1747
 - duplicate, 1748
 - get, 1748
 - hasArray, 1749

- IntArrayBuffer, 1745, 1746
- isReadOnly, 1749
- put, 1749
- setReadOnly, 1750
- slice, 1750
- decaf::internal::nio::LongArrayBuffer, 2071
 - ~LongArrayBuffer, 2073
 - array, 2073
 - arrayOffset, 2074
 - asReadOnlyBuffer, 2074
 - compact, 2074
 - duplicate, 2075
 - get, 2075
 - hasArray, 2076
 - isReadOnly, 2076
 - LongArrayBuffer, 2072, 2073
 - put, 2076, 2077
 - setReadOnly, 2077
 - slice, 2077
- decaf::internal::nio::ShortArrayBuffer, 2915
 - ~ShortArrayBuffer, 2917
 - array, 2917
 - arrayOffset, 2918
 - asReadOnlyBuffer, 2918
 - compact, 2918
 - duplicate, 2919
 - get, 2919
 - hasArray, 2920
 - isReadOnly, 2920
 - put, 2920, 2921
 - setReadOnly, 2921
 - ShortArrayBuffer, 2916, 2917
 - slice, 2921
- decaf::internal::util, 120
- decaf::internal::util::ByteArrayAdapter, 849
 - ~ByteArrayAdapter, 855
 - ByteArrayAdapter, 853–855
 - clear, 856
 - get, 856
 - getByteArray, 856
 - getCapacity, 856
 - getChar, 856
 - getCharArray, 857
 - getCharCapacity, 857
 - getDouble, 857
 - getDoubleArray, 857
 - getDoubleAt, 858
 - getDoubleCapacity, 858
 - getFloat, 858
 - getFloatArray, 859
 - getFloatAt, 859
 - getFloatCapacity, 859
 - getInt, 859
 - getIntArray, 860
 - getIntAt, 860
 - getIntCapacity, 860
 - getLong, 861
 - getLongArray, 861
 - getLongAt, 861
 - getLongCapacity, 862
 - getShort, 862
 - getShortArray, 862
 - getShortAt, 862
 - getShortCapacity, 863
 - operator[], 863
 - put, 863
 - putChar, 864
 - putDouble, 864
 - putDoubleAt, 865
 - putFloat, 865
 - putFloatAt, 865
 - putInt, 866
 - putIntAt, 866
 - putLong, 867
 - putLongAt, 867
 - putShort, 867
 - putShortAt, 868
 - read, 868
 - resize, 869
 - write, 869
- decaf::internal::util::concurrent, 121
- decaf::internal::util::concurrent::ConditionImpl, 1126
 - create, 1126
 - destroy, 1126
 - notify, 1126
 - notifyAll, 1127
 - wait, 1127
- decaf::internal::util::concurrent::MutexImpl, 2391
 - create, 2391
 - destroy, 2391
 - lock, 2391
 - trylock, 2392
 - unlock, 2392
- decaf::internal::util::concurrent::SynchronizableImpl, 3133
 - ~SynchronizableImpl, 3134
 - lock, 3134
 - notify, 3134
 - notifyAll, 3134
 - SynchronizableImpl, 3134
 - tryLock, 3135
 - unlock, 3135
 - wait, 3135, 3136
- decaf::internal::util::concurrent::Transferer, 3267

- decaf::internal::util::concurrent::TransferQueue, 3268
 - ~TransferQueue, 3268
 - transfer, 3269
 - TransferQueue, 3268
- decaf::internal::util::concurrent::TransferStack, 3271
 - ~TransferStack, 3271
 - transfer, 3271, 3272
 - TransferStack, 3271
- decaf::internal::util::HexStringParser, 1713
 - ~HexStringParser, 1713
 - HexStringParser, 1713
 - parse, 1713
 - parseDouble, 1713
 - parseFloat, 1714
- decaf::internal::util::TimerTaskHeap, 3202
 - ~TimerTaskHeap, 3203
 - adjustMinimum, 3203
 - decaf::util::TimerTask, 3201
 - deleteIfCancelled, 3203
 - find, 3203
 - insert, 3203
 - isEmpty, 3203
 - peek, 3203
 - remove, 3204
 - reset, 3204
 - size, 3204
 - TimerTaskHeap, 3203
- decaf::io, 122
- decaf::io::BlockingByteArrayInputStream, 724
 - ~BlockingByteArrayInputStream, 726
 - available, 726
 - BlockingByteArrayInputStream, 725
 - close, 726
 - lock, 726
 - mark, 726
 - markSupported, 727
 - notify, 727
 - notifyAll, 727
 - read, 728
 - reset, 728
 - setByteArray, 729
 - skip, 729
 - tryLock, 729
 - unlock, 730
 - wait, 730, 731
- decaf::io::BufferedInputStream, 809
 - ~BufferedInputStream, 810
 - available, 810
 - BufferedInputStream, 810
 - close, 810
 - mark, 811
 - markSupported, 811
 - read, 811, 812
 - reset, 812
 - skip, 812
- decaf::io::BufferedOutputStream, 814
 - ~BufferedOutputStream, 815
 - BufferedOutputStream, 814, 815
 - close, 815
 - flush, 815
 - write, 815, 816
- decaf::io::ByteArrayInputStream, 892
 - ~ByteArrayInputStream, 894
 - available, 894
 - ByteArrayInputStream, 894
 - close, 894
 - lock, 894
 - mark, 895
 - markSupported, 895
 - notify, 895
 - notifyAll, 895
 - read, 896
 - reset, 896
 - setBuffer, 897
 - setByteArray, 897
 - skip, 897
 - tryLock, 897
 - unlock, 898
 - wait, 898, 899
- decaf::io::ByteArrayOutputStream, 900
 - ~ByteArrayOutputStream, 902
 - ByteArrayOutputStream, 901
 - close, 902
 - flush, 902
 - lock, 902
 - notify, 902
 - notifyAll, 903
 - reset, 903
 - setBuffer, 903
 - size, 903
 - toByteArray, 903
 - toByteArrayRef, 904
 - toString, 904
 - tryLock, 904
 - unlock, 904
 - wait, 904, 905
 - write, 906
 - writeTo, 907
- decaf::io::Closeable, 1019
 - ~Closeable, 1019
 - close, 1019
- decaf::io::DataInputStream, 1379
 - ~DataInputStream, 1381
 - DataInputStream, 1380
 - read, 1381, 1382
 - readBoolean, 1382

- readByte, 1383
- readChar, 1383
- readDouble, 1383
- readFloat, 1383
- readFully, 1384
- readInt, 1385
- readLong, 1385
- readShort, 1385
- readString, 1386
- readUnsignedByte, 1386
- readUnsignedShort, 1386
- readUTF, 1387
- skip, 1387
- decaf::io::DataOutputStream, 1388
 - ~DataOutputStream, 1389
 - buffer, 1394
 - DataOutputStream, 1389
 - size, 1390
 - write, 1390
 - writeBoolean, 1391
 - writeByte, 1391
 - writeBytes, 1391
 - writeChar, 1391
 - writeChars, 1392
 - writeDouble, 1392
 - writeFloat, 1392
 - writeInt, 1393
 - writeLong, 1393
 - writeShort, 1393
 - writeUnsignedShort, 1393
 - writeUTF, 1394
 - written, 1394
- decaf::io::EOFException, 1571
 - ~EOFException, 1572
 - clone, 1573
 - EOFException, 1571, 1572
- decaf::io::FilterInputStream, 1631
 - ~FilterInputStream, 1633
 - available, 1633
 - close, 1633
 - closed, 1639
 - FilterInputStream, 1633
 - inputStream, 1639
 - isClosed, 1633
 - lock, 1634
 - mark, 1634
 - markSupported, 1634
 - mutex, 1639
 - notify, 1634
 - notifyAll, 1635
 - own, 1639
 - read, 1635, 1636
 - reset, 1636
 - skip, 1637
 - tryLock, 1637
 - unlock, 1637
 - wait, 1638, 1639
- decaf::io::FilterOutputStream, 1640
 - ~FilterOutputStream, 1642
 - close, 1642
 - closed, 1646
 - FilterOutputStream, 1641
 - flush, 1642
 - isClosed, 1642
 - lock, 1642
 - mutex, 1646
 - notify, 1643
 - notifyAll, 1643
 - outputStream, 1646
 - own, 1646
 - tryLock, 1643
 - unlock, 1644
 - wait, 1644, 1645
 - write, 1645, 1646
- decaf::io::InputStream, 1740
 - ~InputStream, 1741
 - available, 1741
 - mark, 1741
 - markSupported, 1741
 - read, 1741, 1742
 - reset, 1742
 - skip, 1743
- decaf::io::InterruptedIOException, 1805
 - ~InterruptedIOException, 1806
 - clone, 1807
 - InterruptedIOException, 1805, 1806
- decaf::io::IOException, 1820
 - ~IOException, 1821
 - clone, 1822
 - IOException, 1820, 1821
- decaf::io::OutputStream, 2470
 - ~OutputStream, 2470
 - flush, 2470
 - write, 2470, 2471
- decaf::io::Reader, 2683
 - ~Reader, 2683
 - getInputStream, 2683
 - read, 2683
 - readByte, 2683
 - setInputStream, 2684
- decaf::io::UnsupportedEncodingException, 3300
 - ~UnsupportedEncodingException, 3302
 - clone, 3302
 - UnsupportedEncodingException, 3300, 3301
- decaf::io::UTFDataFormatException, 3353
 - ~UTFDataFormatException, 3355

- clone, 3355
- UTFDataFormatException, 3353, 3354
- decaf::io::Writer, 3404
 - ~Writer, 3404
 - getOutputStream, 3404
 - setOutputStream, 3404
 - write, 3404
 - writeByte, 3405
- decaf::lang, 124
 - operator==, 125
- decaf::lang::Appendable, 626
 - ~Appendable, 626
 - append, 626, 627
- decaf::lang::AtomicRefCounter, 638
 - AtomicRefCounter, 639
 - release, 639
 - swap, 640
- decaf::lang::Boolean, 732
 - ~Boolean, 733
 - _FALSE, 736
 - _TRUE, 736
 - Boolean, 733
 - booleanValue, 733
 - compareTo, 733
 - equals, 734
 - operator<, 734
 - operator==, 735
 - parseBoolean, 735
 - toString, 735, 736
 - valueOf, 736
- decaf::lang::Byte, 840
 - ~Byte, 842
 - Byte, 841
 - byteValue, 842
 - compareTo, 842
 - decode, 843
 - doubleValue, 843
 - equals, 843
 - floatValue, 844
 - intValue, 844
 - longValue, 844
 - MAX_VALUE, 848
 - MIN_VALUE, 848
 - operator<, 844
 - operator==, 845
 - parseByte, 845, 846
 - shortValue, 846
 - SIZE, 848
 - toString, 846
 - valueOf, 847
- decaf::lang::Character, 982
 - byteValue, 984
 - Character, 984
 - compareTo, 984
 - digit, 984
 - doubleValue, 985
 - equals, 985
 - floatValue, 985
 - intValue, 986
 - isDigit, 986
 - isISOControl, 986
 - isLetter, 986
 - isLetterOrDigit, 986
 - isLowerCase, 986
 - isUpperCase, 986
 - isWhitespace, 986
 - longValue, 987
 - MAX_RADIX, 989
 - MAX_VALUE, 989
 - MIN_RADIX, 989
 - MIN_VALUE, 989
 - operator<, 987
 - operator==, 987, 988
 - shortValue, 988
 - SIZE, 989
 - toString, 988
 - valueOf, 988
- decaf::lang::CharSequence, 1014
 - ~CharSequence, 1014
 - charAt, 1014
 - length, 1015
 - subSequence, 1015
 - toString, 1015
- decaf::lang::Comparable, 1083
 - ~Comparable, 1083
 - compareTo, 1083
 - equals, 1084
 - operator<, 1084
 - operator==, 1085
- decaf::lang::Double, 1537
 - ~Double, 1539
 - byteValue, 1539
 - compare, 1539
 - compareTo, 1540
 - Double, 1539
 - doubleToLongBits, 1540
 - doubleToRawLongBits, 1541
 - doubleValue, 1541
 - equals, 1541, 1542
 - floatValue, 1542
 - intValue, 1542
 - isInfinite, 1542
 - isNaN, 1543
 - longBitsToDouble, 1543
 - longValue, 1543
 - MAX_VALUE, 1547
 - MIN_VALUE, 1547
 - NaN, 1547

- NEGATIVE_INFINITY, 1547
- operator<, 1543, 1544
- operator==, 1544
- parseDouble, 1545
- POSITIVE_INFINITY, 1547
- shortValue, 1545
- SIZE, 1547
- toHexString, 1545
- toString, 1546
- valueOf, 1546, 1547
- decaf::lang::DYNAMIC_CAST_TOKEN, 1567
- decaf::lang::Exception, 1574
 - ~Exception, 1576
 - buildMessage, 1577
 - cause, 1580
 - clone, 1577
 - Exception, 1575, 1576
 - getCause, 1577
 - getMessage, 1578
 - getStackTrace, 1578
 - getStackTraceString, 1578
 - initCause, 1578
 - message, 1580
 - operator=, 1578
 - printStackTrace, 1579
 - setMark, 1579
 - setMessage, 1579
 - setStackTrace, 1579
 - stackTrace, 1580
 - what, 1580
- decaf::lang::exceptions, 126
- decaf::lang::exceptions::ClassCastException, 1016
 - ~ClassCastException, 1017
 - ClassCastException, 1016, 1017
 - clone, 1018
- decaf::lang::exceptions::IllegalArgumentException, 1720
 - ~IllegalArgumentException, 1721
 - clone, 1722
 - IllegalArgumentException, 1720, 1721
- decaf::lang::exceptions::IllegalMonitorStateException, 1723
 - ~IllegalMonitorStateException, 1724
 - clone, 1725
 - IllegalMonitorStateException, 1723, 1724
- decaf::lang::exceptions::IllegalStateException, 1726
 - ~IllegalStateException, 1727
 - clone, 1728
 - IllegalStateException, 1726, 1727
- decaf::lang::exceptions::IllegalThreadStateException, 1730
 - ~IllegalThreadStateException, 1731
 - clone, 1732
 - IllegalThreadStateException, 1730, 1731
- decaf::lang::exceptions::IndexOutOfBoundsException, 1737
 - ~IndexOutOfBoundsException, 1738
 - clone, 1739
 - IndexOutOfBoundsException, 1737, 1738
- decaf::lang::exceptions::InterruptedException, 1802
 - ~InterruptedException, 1803
 - clone, 1804
 - InterruptedException, 1802, 1803
- decaf::lang::exceptions::InvalidStateException, 1817
 - ~InvalidStateException, 1818
 - clone, 1819
 - InvalidStateException, 1817, 1818
- decaf::lang::exceptions::NoSuchElementException, 2423
 - ~NoSuchElementException, 2424
 - clone, 2425
 - NoSuchElementException, 2423, 2424
- decaf::lang::exceptions::NullPointerException, 2429
 - ~NullPointerException, 2430
 - clone, 2431
 - NullPointerException, 2429, 2430
- decaf::lang::exceptions::NumberFormatException, 2435
 - ~NumberFormatException, 2437
 - clone, 2437
 - NumberFormatException, 2435, 2436
- decaf::lang::exceptions::RuntimeException, 2819
 - ~RuntimeException, 2820
 - clone, 2821
 - RuntimeException, 2819, 2820
- decaf::lang::exceptions::UnsupportedOperationException, 3303
 - ~UnsupportedOperationException, 3304
 - clone, 3305
 - UnsupportedOperationException, 3303, 3304
- decaf::lang::Float, 1647
 - ~Float, 1649
 - byteValue, 1649
 - compare, 1649
 - compareTo, 1650
 - doubleValue, 1650
 - equals, 1651
 - Float, 1649
 - floatToIntBits, 1651
 - floatToRawIntBits, 1651
 - floatValue, 1652

- intBitsToFloat, 1652
- intValue, 1652
- isInfinite, 1653
- isNaN, 1653
- longValue, 1653
- MAX_VALUE, 1657
- MIN_VALUE, 1657
- NaN, 1657
- NEGATIVE_INFINITY, 1657
- operator<, 1653, 1654
- operator==, 1654
- parseFloat, 1655
- POSITIVE_INFINITY, 1657
- shortValue, 1655
- SIZE, 1657
- toHexString, 1655
- toString, 1656
- valueOf, 1656, 1657
- decaf::lang::Integer, 1762
 - ~Integer, 1765
 - bitCount, 1765
 - byteValue, 1765
 - compareTo, 1765
 - decode, 1766
 - doubleValue, 1766
 - equals, 1766, 1767
 - floatValue, 1767
 - highestOneBit, 1767
 - Integer, 1764
 - intValue, 1767
 - longValue, 1768
 - lowestOneBit, 1768
 - MAX_VALUE, 1775
 - MIN_VALUE, 1775
 - numberOfLeadingZeros, 1768
 - numberOfTrailingZeros, 1768
 - operator<, 1769
 - operator==, 1769, 1770
 - parseInt, 1770
 - reverse, 1771
 - reverseBytes, 1771
 - rotateLeft, 1771
 - rotateRight, 1772
 - shortValue, 1772
 - signum, 1772
 - SIZE, 1776
 - toBinaryString, 1772
 - toHexString, 1773
 - toOctalString, 1773
 - toString, 1774
 - valueOf, 1774, 1775
- decaf::lang::Iterable, 1830
 - ~Iterable, 1830
 - iterator, 1830
- decaf::lang::Long, 2057
 - ~Long, 2060
 - bitCount, 2060
 - byteValue, 2060
 - compareTo, 2060
 - decode, 2061
 - doubleValue, 2061
 - equals, 2061, 2062
 - floatValue, 2062
 - highestOneBit, 2062
 - intValue, 2062
 - Long, 2059
 - longValue, 2062
 - lowestOneBit, 2063
 - MAX_VALUE, 2070
 - MIN_VALUE, 2070
 - numberOfLeadingZeros, 2063
 - numberOfTrailingZeros, 2063
 - operator<, 2064
 - operator==, 2064
 - parseLong, 2065
 - reverse, 2066
 - reverseBytes, 2066
 - rotateLeft, 2066
 - rotateRight, 2066
 - shortValue, 2067
 - signum, 2067
 - SIZE, 2070
 - toBinaryString, 2067
 - toHexString, 2068
 - toOctalString, 2068
 - toString, 2068, 2069
 - valueOf, 2069, 2070
- decaf::lang::Math, 2126
 - ~Math, 2128
 - abs, 2128, 2129
 - ceil, 2129
 - E, 2140
 - floor, 2130
 - Math, 2128
 - max, 2130, 2131
 - min, 2132, 2133
 - PI, 2140
 - pow, 2134
 - random, 2134
 - round, 2135
 - signum, 2135, 2136
 - sqrt, 2137
 - toDegrees, 2140
 - toRadians, 2140
- decaf::lang::Number, 2432
 - ~Number, 2432
 - byteValue, 2432
 - doubleValue, 2433

- floatValue, 2433
- intValue, 2433
- longValue, 2433
- shortValue, 2434
- decaf::lang::Pointer, 2497
 - ~Pointer, 2500
 - CounterType, 2499
 - dynamicCast, 2500
 - get, 2500
 - operator*, 2501
 - operator->, 2501
 - operator=, 2502
 - operator==, 2502, 2503
 - Pointer, 2499, 2500
 - PointerType, 2499
 - ReferenceType, 2499
 - release, 2502
 - reset, 2502
 - staticCast, 2502
 - swap, 2503
- decaf::lang::PointerComparator, 2504
 - compare, 2504
 - operator(), 2504
- decaf::lang::Runnable, 2816
 - ~Runnable, 2816
 - run, 2816
- decaf::lang::Runtime, 2817
 - ~Runtime, 2817
 - getRuntime, 2817
 - initializeRuntime, 2817
 - shutdownRuntime, 2818
- decaf::lang::Short, 2906
 - ~Short, 2908
 - byteValue, 2908
 - compareTo, 2908
 - decode, 2909
 - doubleValue, 2909
 - equals, 2909
 - floatValue, 2909
 - intValue, 2910
 - longValue, 2910
 - MAX_VALUE, 2914
 - MIN_VALUE, 2914
 - operator<, 2910
 - operator==, 2911
 - parseShort, 2911
 - reverseBytes, 2912
 - Short, 2907
 - shortValue, 2912
 - SIZE, 2914
 - toString, 2912
 - valueOf, 2913
- decaf::lang::STATIC_CAST_TOKEN, 3015
- decaf::lang::System, 3145
 - ~System, 3145
 - availableProcessors, 3145
 - currentTimeMillis, 3146
 - getenv, 3146
 - nanoTime, 3146
 - setenv, 3147
 - System, 3145
 - unsetenv, 3147
- decaf::lang::ThreadGroup, 3174
 - ~ThreadGroup, 3174
 - ThreadGroup, 3174
- decaf::lang::Throwable, 3181
 - ~Throwable, 3182
 - clone, 3182
 - getCause, 3182
 - getMessage, 3183
 - getStackTrace, 3183
 - getStackTraceString, 3183
 - initCause, 3183
 - printStackTrace, 3184
 - setMark, 3184
 - Throwable, 3182
- decaf::net, 127
- decaf::net::BindException, 721
 - ~BindException, 722
 - BindException, 721, 722
 - clone, 723
- decaf::net::BufferedSocket, 817
 - ~BufferedSocket, 818
 - BufferedSocket, 818
 - close, 818
 - connect, 818
 - getInputStream, 819
 - getKeepAlive, 819
 - getOutputStream, 819
 - getReceiveBufferSize, 819
 - getReuseAddress, 820
 - getSendBufferSize, 820
 - getSoLinger, 820
 - getSoTimeout, 821
 - isConnected, 821
 - setKeepAlive, 821
 - setReceiveBufferSize, 821
 - setReuseAddress, 822
 - setSendBufferSize, 822
 - setSoLinger, 822
 - setSoTimeout, 823
- decaf::net::ConnectException, 1128
 - ~ConnectException, 1129
 - clone, 1130
 - ConnectException, 1128, 1129
- decaf::net::HttpRetryException, 1717
 - ~HttpRetryException, 1718
 - clone, 1719

- HttpRetryException, 1717, 1718
- decaf::net::MalformedURLException, 2091
 - ~MalformedURLException, 2092
 - clone, 2093
 - MalformedURLException, 2091, 2092
- decaf::net::NoRouteToHostException, 2417
 - ~NoRouteToHostException, 2418
 - clone, 2419
 - NoRouteToHostException, 2417, 2418
- decaf::net::PortUnreachableException, 2522
 - ~PortUnreachableException, 2523
 - clone, 2524
 - PortUnreachableException, 2522, 2523
- decaf::net::ProtocolException, 2667
 - ~ProtocolException, 2668
 - clone, 2669
 - ProtocolException, 2667, 2668
- decaf::net::ServerSocket, 2837
 - ~ServerSocket, 2837
 - accept, 2838
 - bind, 2838
 - close, 2838
 - isBound, 2838
 - ServerSocket, 2837
 - SocketAddress, 2837
 - SocketHandle, 2837
- decaf::net::Socket, 2964
 - ~Socket, 2965
 - connect, 2965
 - getInputStream, 2965
 - getKeepAlive, 2966
 - getOutputStream, 2966
 - getReceiveBufferSize, 2966
 - getReuseAddress, 2966
 - getSendBufferSize, 2967
 - getSoLinger, 2967
 - getSoTimeout, 2967
 - isConnected, 2968
 - setKeepAlive, 2968
 - setReceiveBufferSize, 2968
 - setReuseAddress, 2968
 - setSendBufferSize, 2969
 - setSoLinger, 2969
 - setSoTimeout, 2969
 - SocketAddress, 2965
 - SocketHandle, 2965
- decaf::net::SocketError, 2971
 - getErrorCode, 2971
 - getErrorString, 2971
- decaf::net::SocketException, 2972
 - ~SocketException, 2973
 - clone, 2973
 - SocketException, 2972, 2973
- decaf::net::SocketFactory, 2975
 - ~SocketFactory, 2975
 - createSocket, 2975
- decaf::net::SocketInputStream, 2977
 - ~SocketInputStream, 2978
 - available, 2979
 - close, 2979
 - lock, 2979
 - mark, 2979
 - markSupported, 2979
 - notify, 2980
 - notifyAll, 2980
 - read, 2980, 2981
 - reset, 2981
 - skip, 2981
 - SocketInputStream, 2978
 - tryLock, 2982
 - unlock, 2982
 - wait, 2982, 2983
- decaf::net::SocketOutputStream, 2984
 - ~SocketOutputStream, 2985
 - close, 2985
 - flush, 2985
 - lock, 2986
 - notify, 2986
 - notifyAll, 2986
 - SocketOutputStream, 2985
 - tryLock, 2986
 - unlock, 2987
 - wait, 2987, 2988
 - write, 2988, 2989
- decaf::net::SocketTimeoutException, 2990
 - ~SocketTimeoutException, 2991
 - clone, 2992
 - SocketTimeoutException, 2990, 2991
- decaf::net::TcpSocket, 3152
 - ~TcpSocket, 3154
 - checkResult, 3154
 - close, 3154
 - connect, 3154
 - getInputStream, 3155
 - getKeepAlive, 3155
 - getOutputStream, 3155
 - getReceiveBufferSize, 3155
 - getReuseAddress, 3156
 - getSendBufferSize, 3156
 - getSocketHandle, 3156
 - getSoLinger, 3156
 - getSoTimeout, 3156
 - getTcpNoDelay, 3157
 - isConnected, 3157
 - setKeepAlive, 3157
 - setReceiveBufferSize, 3157
 - setReuseAddress, 3158
 - setSendBufferSize, 3158

- setSoLinger, 3158
 - setSoTimeout, 3158
 - setTcpNoDelay, 3159
 - TcpSocket, 3153
- decaf::net::UnknownHostException, 3294
 - ~UnknownHostException, 3295
 - clone, 3296
 - UnknownHostException, 3294, 3295
- decaf::net::UnknownServiceException, 3297
 - ~UnknownServiceException, 3298
 - clone, 3299
 - UnknownServiceException, 3297, 3298
- decaf::net::URI, 3308
 - ~URI, 3311
 - compareTo, 3312
 - create, 3312
 - equals, 3312
 - getAuthority, 3312
 - getFragment, 3312
 - getHost, 3312
 - getPath, 3313
 - getPort, 3313
 - getQuery, 3313
 - getRawAuthority, 3313
 - getRawFragment, 3313
 - getRawPath, 3313
 - getRawQuery, 3314
 - getRawSchemeSpecificPart, 3314
 - getRawUserInfo, 3314
 - getScheme, 3314
 - getSchemeSpecificPart, 3314
 - getUserInfo, 3314
 - isAbsolute, 3315
 - isOpaque, 3315
 - normalize, 3315
 - operator<, 3315
 - operator==, 3316
 - parseServerAuthority, 3316
 - relativize, 3316
 - resolve, 3317
 - toString, 3318
 - toURL, 3318
 - URI, 3310, 3311
- decaf::net::URISyntaxException, 3335
 - ~URISyntaxException, 3337
 - clone, 3337
 - getIndex, 3337
 - getInput, 3338
 - getReason, 3338
 - URISyntaxException, 3335–3337
- decaf::net::URL, 3347
 - ~URL, 3348
 - URL, 3348
- decaf::net::URLDecoder, 3349
 - ~URLDecoder, 3349
 - decode, 3349
- decaf::net::URLEncoder, 3350
 - ~URLEncoder, 3350
 - encode, 3350
- decaf::nio, 128
- decaf::nio::Buffer, 803
 - ~Buffer, 805
 - _capacity, 808
 - _limit, 808
 - _mark, 808
 - _markSet, 808
 - _position, 808
 - Buffer, 805
 - capacity, 805
 - clear, 805
 - flip, 806
 - hasRemaining, 806
 - isReadOnly, 806
 - limit, 806, 807
 - mark, 807
 - position, 807
 - remaining, 807
 - reset, 808
 - rewind, 808
- decaf::nio::BufferOverflowException, 834
 - ~BufferOverflowException, 835
 - BufferOverflowException, 834, 835
 - clone, 836
- decaf::nio::BufferUnderflowException, 837
 - ~BufferUnderflowException, 838
 - BufferUnderflowException, 837, 838
 - clone, 839
- decaf::nio::ByteBuffer, 913
 - ~ByteBuffer, 918
 - allocate, 918
 - array, 918
 - arrayOffset, 919
 - asCharBuffer, 919
 - asDoubleBuffer, 919
 - asFloatBuffer, 920
 - asIntBuffer, 920
 - asLongBuffer, 920
 - asReadOnlyBuffer, 921
 - asShortBuffer, 921
 - ByteBuffer, 918
 - compact, 921
 - compareTo, 922
 - duplicate, 922
 - equals, 922
 - get, 922, 923
 - getChar, 924
 - getDouble, 924, 925
 - getFloat, 925

- getInt, 926
- getLong, 926, 927
- getShort, 927
- hasArray, 928
- isReadOnly, 928
- operator<, 928
- operator==, 928
- put, 929, 930
- putChar, 931
- putDouble, 932
- putFloat, 932, 933
- putInt, 933, 934
- putLong, 934
- putShort, 935
- slice, 936
- toString, 936
- wrap, 936
- decaf::nio::CharBuffer, 998
 - ~CharBuffer, 1001
 - allocate, 1001
 - append, 1001, 1002
 - array, 1003
 - arrayOffset, 1003
 - asReadOnlyBuffer, 1003
 - charAt, 1004
 - CharBuffer, 1001
 - compact, 1004
 - compareTo, 1004
 - duplicate, 1005
 - equals, 1005
 - get, 1005, 1006
 - hasArray, 1007
 - length, 1007
 - operator<, 1007
 - operator==, 1007
 - put, 1008–1010
 - read, 1011
 - slice, 1011
 - subSequence, 1011
 - toString, 1012
 - wrap, 1012
- decaf::nio::DoubleBuffer, 1556
 - ~DoubleBuffer, 1558
 - allocate, 1558
 - array, 1558
 - arrayOffset, 1559
 - asReadOnlyBuffer, 1559
 - compact, 1559
 - compareTo, 1560
 - DoubleBuffer, 1558
 - duplicate, 1560
 - equals, 1560
 - get, 1561, 1562
 - hasArray, 1562
 - operator<, 1562
 - operator==, 1563
 - put, 1563–1565
 - slice, 1565
 - toString, 1565
 - wrap, 1565, 1566
- decaf::nio::FloatBuffer, 1665
 - ~FloatBuffer, 1667
 - allocate, 1667
 - array, 1667
 - arrayOffset, 1668
 - asReadOnlyBuffer, 1668
 - compact, 1668
 - compareTo, 1669
 - duplicate, 1669
 - equals, 1669
 - FloatBuffer, 1667
 - get, 1669–1671
 - hasArray, 1671
 - operator<, 1671
 - operator==, 1671
 - put, 1672, 1673
 - slice, 1674
 - toString, 1674
 - wrap, 1674, 1675
- decaf::nio::IntBuffer, 1751
 - ~IntBuffer, 1753
 - allocate, 1753
 - array, 1753
 - arrayOffset, 1754
 - asReadOnlyBuffer, 1754
 - compact, 1754
 - compareTo, 1755
 - duplicate, 1755
 - equals, 1755
 - get, 1755–1757
 - hasArray, 1757
 - IntBuffer, 1753
 - operator<, 1757
 - operator==, 1757
 - put, 1758, 1759
 - slice, 1760
 - toString, 1760
 - wrap, 1760, 1761
- decaf::nio::InvalidMarkException, 1813
 - ~InvalidMarkException, 1814
 - clone, 1815
 - InvalidMarkException, 1813, 1814
- decaf::nio::LongBuffer, 2079
 - ~LongBuffer, 2081
 - allocate, 2081
 - array, 2081
 - arrayOffset, 2082
 - asReadOnlyBuffer, 2082

- compact, 2082
- compareTo, 2083
- duplicate, 2083
- equals, 2083
- get, 2084, 2085
- hasArray, 2085
- LongBuffer, 2081
- operator<, 2085
- operator==, 2086
- put, 2086–2088
- slice, 2088
- toString, 2088
- wrap, 2089
- decaf::nio::ReadOnlyBufferException, 2685
 - ~ReadOnlyBufferException, 2686
 - clone, 2687
 - ReadOnlyBufferException, 2685, 2686
- decaf::nio::ShortBuffer, 2923
 - ~ShortBuffer, 2925
 - allocate, 2925
 - array, 2925
 - arrayOffset, 2926
 - asReadOnlyBuffer, 2926
 - compact, 2926
 - compareTo, 2927
 - duplicate, 2927
 - equals, 2927
 - get, 2928, 2929
 - hasArray, 2929
 - operator<, 2929
 - operator==, 2930
 - put, 2930, 2931
 - ShortBuffer, 2925
 - slice, 2932
 - toString, 2932
 - wrap, 2932, 2933
- decaf::security, 129
- decaf::security::auth, 130
- decaf::security::auth::x500, 131
- decaf::security::auth::x500::X500Principal, 3406
 - ~X500Principal, 3406
 - getEncoded, 3406
 - getName, 3406
 - hashCode, 3406
- decaf::security::cert, 132
- decaf::security::cert::Certificate, 968
 - ~Certificate, 969
 - equals, 969
 - getEncoded, 969
 - getPublicKey, 969
 - getType, 970
 - toString, 970
 - verify, 970
- decaf::security::cert::CertificateEncodingException, 972
 - ~CertificateEncodingException, 973
 - CertificateEncodingException, 972, 973
 - clone, 973
- decaf::security::cert::CertificateException, 974
 - ~CertificateException, 975
 - CertificateException, 974, 975
 - clone, 975
- decaf::security::cert::CertificateExpiredException, 976
 - ~CertificateExpiredException, 977
 - CertificateExpiredException, 976, 977
 - clone, 977
- decaf::security::cert::CertificateNotYetValidException, 978
 - ~CertificateNotYetValidException, 979
 - CertificateNotYetValidException, 978, 979
 - clone, 979
- decaf::security::cert::CertificateParsingException, 980
 - ~CertificateParsingException, 981
 - CertificateParsingException, 980, 981
 - clone, 981
- decaf::security::cert::X509Certificate, 3407
 - ~X509Certificate, 3407
 - checkValidity, 3407
 - getBasicConstraints, 3408
 - getIssuerUniqueID, 3408
 - getIssuerX500Principal, 3408
 - getKeyUsage, 3408
 - getNotAfter, 3408
 - getNotBefore, 3408
 - getSigAlgName, 3408
 - getSigAlgOID, 3408
 - getSigAlgParams, 3409
 - getSignature, 3409
 - getSubjectUniqueID, 3409
 - getSubjectX500Principal, 3409
 - getTBSCertificate, 3409
 - getVersion, 3409
- decaf::security::GeneralSecurityException, 1706
 - ~GeneralSecurityException, 1707
 - clone, 1708
 - GeneralSecurityException, 1706, 1707
- decaf::security::InvalidKeyException, 1810
 - ~InvalidKeyException, 1811
 - clone, 1812
 - InvalidKeyException, 1810, 1811
- decaf::security::Key, 1953
 - ~Key, 1954
 - getAlgorithm, 1954
 - getEncoded, 1954
 - getFormat, 1954

- decaf::security::KeyException, 1955
 - ~KeyException, 1956
 - clone, 1957
 - KeyException, 1955, 1956
- decaf::security::NoSuchAlgorithmException, 2420
 - ~NoSuchAlgorithmException, 2421
 - clone, 2422
 - NoSuchAlgorithmException, 2420, 2421
- decaf::security::NoSuchProviderException, 2426
 - ~NoSuchProviderException, 2427
 - clone, 2428
 - NoSuchProviderException, 2426, 2427
- decaf::security::Principal, 2569
 - ~Principal, 2569
 - equals, 2569
 - getName, 2569
- decaf::security::PublicKey, 2670
 - ~PublicKey, 2670
- decaf::security::SignatureException, 2957
 - ~SignatureException, 2958
 - clone, 2959
 - SignatureException, 2957, 2958
- decaf::security_provider, 133
- decaf::security_provider::SecurityProvider, 2822
 - ~SecurityProvider, 2822
 - createX500Principal, 2822
- decaf::security_provider::SecurityProviderMap, 2823
 - getInstance, 2823
 - getSecurityProviderNames, 2823
 - lookup, 2824
 - registerSecurityProvider, 2824
 - unregisterSecurityProvider, 2824
- decaf::security_provider::SecurityProviderRegistrar, 2825
 - ~SecurityProviderRegistrar, 2825
 - getProvider, 2826
 - SecurityProviderRegistrar, 2825
- decaf::security_provider::unix, 134
- decaf::security_provider::unix::openssl, 135
- decaf::security_provider::unix::openssl::OpenSSLX500Principal, 2439
 - ~OpenSSLX500Principal, 2440
 - equals, 2440
 - getEncoded, 2440
 - getName, 2440
 - getX509Name, 2441
 - OpenSSLX500Principal, 2439
 - toString, 2441
- decaf::security_provider::unix::openssl::OpenSSLX509Certificate, 2442
 - ~OpenSSLX509Certificate, 2443
 - checkValidity, 2443
 - equals, 2443
 - getBasicConstraints, 2443
 - getEncoded, 2444
 - getIssuerUniqueID, 2444
 - getIssuerX500Principal, 2444
 - getKeyUsage, 2444
 - getNotAfter, 2444
 - getNotBefore, 2444
 - getPublicKey, 2444, 2445
 - getSigAlgName, 2445
 - getSigAlgOID, 2445
 - getSigAlgParams, 2445
 - getSignature, 2445
 - getSubjectUniqueID, 2445
 - getSubjectX500Principal, 2446
 - getTBSertificate, 2446
 - getType, 2446
 - getVersion, 2446
 - toString, 2446
 - verify, 2446, 2447
- decaf::util, 136
- decaf::util::AbstractCollection, 147
 - ~AbstractCollection, 150
 - add, 150
 - addAll, 150
 - clear, 151
 - contains, 152
 - containsAll, 152
 - copy, 153
 - equals, 153
 - isEmpty, 153
 - lock, 154
 - mutex, 159
 - notify, 154
 - notifyAll, 154
 - operator=, 154
 - remove, 155
 - removeAll, 155
 - retainAll, 156
 - toArray, 157
 - tryLock, 157
 - wait, 157, 158
- decaf::util::AbstractList, 160
 - ~AbstractList, 160
- decaf::util::AbstractMap, 161
 - ~AbstractMap, 161
- decaf::util::AbstractQueue, 162
 - ~AbstractQueue, 163
 - AbstractQueue, 163
 - addAll, 163
 - clear, 164

- element, 164
 - remove, 164
- decaf::util::AbstractSequentialList, 166
 - ~AbstractSequentialList, 166
- decaf::util::AbstractSet, 167
 - ~AbstractSet, 167
 - removeAll, 167
- decaf::util::Collection, 1054
 - ~Collection, 1056
 - add, 1056
 - addAll, 1056
 - clear, 1057
 - contains, 1058
 - containsAll, 1058
 - equals, 1059
 - isEmpty, 1059
 - remove, 1060
 - removeAll, 1060
 - retainAll, 1061
 - size, 1061
 - toArray, 1062
- decaf::util::Comparator, 1086
 - ~Comparator, 1086
 - compare, 1086
 - operator(), 1087
- decaf::util::comparators, 138
- decaf::util::comparators::Less, 1981
 - ~Less, 1981
 - compare, 1981
 - Less, 1981
 - operator(), 1982
- decaf::util::concurrent, 139
- decaf::util::concurrent::atomic, 141
- decaf::util::concurrent::atomic::AtomicBoolean, 630
 - ~AtomicBoolean, 630
 - AtomicBoolean, 630
 - compareAndSet, 631
 - get, 631
 - getAndSet, 631
 - set, 631
 - toString, 631
- decaf::util::concurrent::atomic::AtomicInteger, 633
 - ~AtomicInteger, 634
 - addAndGet, 634
 - AtomicInteger, 634
 - compareAndSet, 635
 - decrementAndGet, 635
 - doubleValue, 635
 - floatValue, 635
 - get, 635
 - getAndAdd, 636
 - getAndDecrement, 636
 - getAndIncrement, 636
 - getAndSet, 636
 - incrementAndGet, 636
 - intValue, 637
 - longValue, 637
 - set, 637
 - toString, 637
- decaf::util::concurrent::atomic::AtomicReference, 641
 - ~AtomicReference, 642
 - AtomicReference, 642
 - compareAndSet, 642
 - get, 642
 - getAndSet, 642
 - set, 642
 - toString, 643
- decaf::util::concurrent::BrokenBarrierException, 742
 - ~BrokenBarrierException, 743
 - BrokenBarrierException, 742, 743
 - clone, 744
- decaf::util::concurrent::Callable, 964
 - ~Callable, 964
 - call, 964
- decaf::util::concurrent::CancellationException, 965
 - ~CancellationException, 966
 - CancellationException, 965, 966
 - clone, 967
- decaf::util::concurrent::ConcurrentMap, 1097
 - ~ConcurrentMap, 1098
 - putIfAbsent, 1098
 - remove, 1098
 - replace, 1099, 1100
- decaf::util::concurrent::ConcurrentStlMap, 1102
 - ~ConcurrentStlMap, 1106
 - clear, 1106
 - ConcurrentStlMap, 1106
 - containsKey, 1107
 - containsValue, 1107
 - copy, 1107, 1108
 - equals, 1108
 - get, 1108, 1109
 - isEmpty, 1109
 - keySet, 1109
 - lock, 1110
 - notify, 1110
 - notifyAll, 1110
 - put, 1110
 - putAll, 1111
 - putIfAbsent, 1111
 - remove, 1112, 1113
 - replace, 1113, 1114
 - size, 1114

- tryLock, 1114
- unlock, 1115
- values, 1115
- wait, 1115, 1116
- decaf::util::concurrent::ConditionHandle, 1124
 - ~ConditionHandle, 1124
 - condition, 1124
 - ConditionHandle, 1124
 - criticalSection, 1124
 - generation, 1124
 - mutex, 1124
 - numWaiting, 1124
 - numWake, 1124
 - semaphore, 1124
- decaf::util::concurrent::CountDownLatch, 1354
 - ~CountDownLatch, 1354
 - await, 1354
 - countDown, 1355
 - CountDownLatch, 1354
 - getCount, 1355
- decaf::util::concurrent::Delayed, 1477
 - ~Delayed, 1477
 - getDelay, 1477
- decaf::util::concurrent::ExecutionException, 1605
 - ~ExecutionException, 1606
 - clone, 1607
 - ExecutionException, 1605, 1606
- decaf::util::concurrent::Executor, 1608
 - ~Executor, 1609
 - execute, 1609
- decaf::util::concurrent::ExecutorService, 1610
 - ~ExecutorService, 1611
 - awaitTermination, 1611
- decaf::util::concurrent::Future, 1701
 - ~Future, 1702
 - cancel, 1702
 - get, 1702
 - isCancelled, 1703
 - isDone, 1703
- decaf::util::concurrent::Lock, 2023
 - ~Lock, 2023
 - isLocked, 2024
 - Lock, 2023
 - lock, 2024
 - unlock, 2024
- decaf::util::concurrent::locks, 142
- decaf::util::concurrent::locks::Condition, 1118
 - ~Condition, 1120
 - await, 1120
 - awaitNanos, 1121
 - awaitUninterruptibly, 1122
 - awaitUntil, 1123
 - signal, 1123
 - signalAll, 1123
- decaf::util::concurrent::locks::Lock, 2017
 - ~Lock, 2018
 - lock, 2018
 - lockInterruptibly, 2019
 - newCondition, 2019
 - tryLock, 2020, 2021
 - unlock, 2021
- decaf::util::concurrent::locks::LockSupport, 2025
 - ~LockSupport, 2026
 - park, 2026
 - parkNanos, 2026
 - parkUntil, 2027
 - unpark, 2027
- decaf::util::concurrent::locks::ReadWriteLock, 2688
 - ~ReadWriteLock, 2689
 - readLock, 2689
 - writeLock, 2689
- decaf::util::concurrent::locks::ReentrantLock, 2692
 - ~ReentrantLock, 2693
 - getHoldCount, 2693
 - isFair, 2694
 - isHeldByCurrentThread, 2694
 - isLocked, 2694
 - lock, 2694
 - lockInterruptibly, 2695
 - newCondition, 2695
 - ReentrantLock, 2693
 - toString, 2696
 - tryLock, 2696, 2697
 - unlock, 2698
- decaf::util::concurrent::Mutex, 2385
 - ~Mutex, 2386
 - lock, 2386
 - Mutex, 2386
 - notify, 2386
 - notifyAll, 2386
 - tryLock, 2387
 - unlock, 2387
 - wait, 2388, 2389
- decaf::util::concurrent::MutexHandle, 2390
 - ~MutexHandle, 2390
 - lock_count, 2390
 - lock_owner, 2390
 - mutex, 2390
 - MutexHandle, 2390
- decaf::util::concurrent::PooledThread, 2518
 - ~PooledThread, 2518
 - getPooledThreadListener, 2518
 - isBusy, 2519
 - PooledThread, 2518

- run, 2519
- setPooledThreadListener, 2519
- stop, 2519
- decaf::util::concurrent::PooledThreadListener, 2520
 - ~PooledThreadListener, 2520
 - onTaskCompleted, 2520
 - onTaskException, 2521
 - onTaskStarted, 2521
- decaf::util::concurrent::RejectedExecutionException, 2699
 - ~RejectedExecutionException, 2700
 - clone, 2701
 - RejectedExecutionException, 2699, 2700
- decaf::util::concurrent::RejectedExecutionHandler, 2702
 - ~RejectedExecutionHandler, 2702
- decaf::util::concurrent::Semaphore, 2827
 - ~Semaphore, 2830
 - acquire, 2830
 - acquireUninterruptibly, 2831
 - availablePermits, 2831
 - drainPermits, 2832
 - isFair, 2832
 - release, 2832
 - Semaphore, 2829
 - toString, 2833
 - tryAcquire, 2833–2835
- decaf::util::concurrent::Synchronizable, 3122
 - ~Synchronizable, 3123
 - lock, 3123
 - notify, 3124
 - notifyAll, 3125
 - tryLock, 3126
 - unlock, 3127
 - wait, 3128, 3130, 3131
- decaf::util::concurrent::SynchronousQueue, 3138
 - ~SynchronousQueue, 3140
 - clear, 3140
 - contains, 3140
 - containsAll, 3140
 - drainTo, 3140
 - equals, 3140
 - isEmpty, 3141
 - iterator, 3141
 - offer, 3141
 - peek, 3142
 - poll, 3142
 - put, 3142
 - remainingCapacity, 3143
 - remove, 3143
 - removeAll, 3143
 - retainAll, 3143
 - size, 3143
 - SynchronousQueue, 3140
 - take, 3143
 - toArray, 3143
- decaf::util::concurrent::TaskListener, 3149
 - ~TaskListener, 3149
 - onTaskComplete, 3149
 - onTaskException, 3149
- decaf::util::concurrent::ThreadFactory, 3172
 - ~ThreadFactory, 3172
 - newThread, 3172
- decaf::util::concurrent::ThreadPool, 3175
 - ~ThreadPool, 3177
 - DEFAULT_MAX_BLOCK_SIZE, 3180
 - DEFAULT_MAX_POOL_SIZE, 3180
 - deQueueTask, 3177
 - getBacklog, 3177
 - getBlockSize, 3177
 - getFreeThreadCount, 3177
 - getInstance, 3178
 - getMaxThreads, 3178
 - getPoolSize, 3178
 - onTaskCompleted, 3178
 - onTaskException, 3178
 - onTaskStarted, 3179
 - queueTask, 3179
 - reserve, 3179
 - setBlockSize, 3179
 - setMaxThreads, 3179
 - Task, 3177
 - ThreadPool, 3177
- decaf::util::concurrent::TimeoutException, 3185
 - ~TimeoutException, 3186
 - clone, 3187
 - TimeoutException, 3185, 3186
- decaf::util::concurrent::TimeUnit, 3205
 - ~TimeUnit, 3207
 - compareTo, 3207
 - convert, 3208
 - DAYS, 3213
 - equals, 3208
 - HOURS, 3213
 - MICROSECONDS, 3213
 - MILLISECONDS, 3213
 - MINUTES, 3213
 - NANOSECONDS, 3213
 - operator<, 3208
 - operator==, 3208
 - SECONDS, 3213
 - sleep, 3209
 - timedJoin, 3209
 - timedWait, 3209
 - TimeUnit, 3207
 - toDays, 3210

- toHours, 3210
- toMicros, 3211
- toMillis, 3211
- toMinutes, 3211
- toNanos, 3212
- toSeconds, 3212
- toString, 3212
- valueOf, 3213
- values, 3214
- decaf::util::Date, 1467
 - ~Date, 1468
 - after, 1468
 - before, 1468
 - compareTo, 1469
 - Date, 1468
 - equals, 1469
 - getTime, 1469
 - operator<, 1469
 - operator=, 1469
 - operator==, 1470
 - setTime, 1470
 - toString, 1470
- decaf::util::Iterator, 1832
 - ~Iterator, 1832
 - hasNext, 1832
 - next, 1832
 - remove, 1833
- decaf::util::List, 1984
 - ~List, 1985
 - add, 1985
 - addAll, 1985
 - get, 1986
 - indexOf, 1986
 - lastIndexOf, 1987
 - listIterator, 1987, 1988
 - remove, 1988
 - set, 1989
- decaf::util::ListIterator, 1990
 - ~ListIterator, 1991
 - add, 1991
 - hasPrevious, 1991
 - nextIndex, 1991
 - previous, 1991
 - previousIndex, 1992
 - set, 1992
- decaf::util::logging, 143
 - Debug, 144
 - Error, 144
 - Fatal, 144
 - Info, 144
 - Level, 143
 - Markblock, 144
 - Null, 144
 - Off, 144
 - Throwing, 144
 - Warn, 144
- decaf::util::logging::Filter, 1630
 - ~Filter, 1630
 - isLoggable, 1630
- decaf::util::logging::Formatter, 1699
 - ~Formatter, 1699
 - format, 1699
 - formatMessage, 1699
 - getHead, 1700
 - getTail, 1700
- decaf::util::logging::Handler, 1709
 - ~Handler, 1710
 - flush, 1710
 - getFilter, 1710
 - getFormatter, 1710
 - getLevel, 1710
 - isLoggable, 1710
 - publish, 1711
 - setFilter, 1711
 - setFormatter, 1711
 - setLevel, 1711
- decaf::util::logging::Logger, 2028
 - ~Logger, 2030
 - addHandler, 2030
 - debug, 2030
 - entry, 2030
 - error, 2030
 - exit, 2031
 - fatal, 2031
 - getAnonymousLogger, 2031
 - getFilter, 2032
 - getLevel, 2032
 - getLogger, 2032
 - getName, 2032
 - getParentHandlers, 2032
 - info, 2033
 - isLoggable, 2033
 - log, 2033, 2034
 - Logger, 2029
 - removeHandler, 2034
 - setFilter, 2035
 - setLevel, 2035
 - setParentHandlers, 2035
 - warn, 2035
- decaf::util::logging::LoggerHierarchy, 2037
 - ~LoggerHierarchy, 2037
 - LoggerHierarchy, 2037
- decaf::util::logging::LogManager, 2045
 - ~LogManager, 2047
 - addPropertyChangeListener, 2047
 - destroy, 2047
 - getInstance, 2048
 - getLogger, 2048

- getLoggerNames, 2048
- getProperties, 2048
- getProperty, 2048
- LogManager, 2047
- operator=, 2049
- removePropertyChangeListener, 2049
- returnInstance, 2049
- setProperties, 2049
- decaf::util::logging::LogRecord, 2050
 - ~LogRecord, 2051
 - getLevel, 2051
 - getLoggerName, 2051
 - getMessage, 2051
 - getSourceFile, 2051
 - getSourceFunction, 2051
 - getSourceLine, 2052
 - getTimestamp, 2052
 - getTreadId, 2052
 - LogRecord, 2051
 - setLevel, 2052
 - setLoggerName, 2052
 - setMessage, 2052
 - setSourceFile, 2053
 - setSourceFunction, 2053
 - setSourceLine, 2053
 - setTimestamp, 2053
 - setTreadId, 2053
- decaf::util::logging::LogWriter, 2055
 - ~LogWriter, 2055
 - destroy, 2055
 - getInstance, 2055
 - log, 2055, 2056
 - LogWriter, 2055
 - returnInstance, 2056
- decaf::util::logging::MarkBlockLogger, 2117
 - ~MarkBlockLogger, 2117
 - MarkBlockLogger, 2117
- decaf::util::logging::PropertiesChangeListener, 2666
 - ~PropertiesChangeListener, 2666
 - onPropertyChanged, 2666
- decaf::util::logging::SimpleFormatter, 2960
 - ~SimpleFormatter, 2960
 - format, 2960
 - formatMessage, 2960
 - getHead, 2961
 - getTail, 2961
 - SimpleFormatter, 2960
- decaf::util::logging::SimpleLogger, 2962
 - ~SimpleLogger, 2962
 - debug, 2963
 - error, 2963
 - fatal, 2963
 - info, 2963
 - log, 2963
 - mark, 2963
 - SimpleLogger, 2962
 - warn, 2963
- decaf::util::logging::StreamHandler, 3077
 - ~StreamHandler, 3078
 - close, 3078
 - flush, 3078
 - getFilter, 3078
 - getFormatter, 3078
 - getLevel, 3079
 - getOutputStream, 3079
 - isLoggable, 3079
 - publish, 3079
 - setFilter, 3079
 - setFormatter, 3080
 - setLevel, 3080
 - StreamHandler, 3078
- decaf::util::Map, 2094
 - ~Map, 2095
 - clear, 2095
 - containsKey, 2096
 - containsValue, 2097
 - copy, 2097
 - equals, 2098
 - get, 2098, 2099
 - isEmpty, 2100
 - keySet, 2101
 - Map, 2095
 - put, 2101
 - putAll, 2102
 - remove, 2103
 - size, 2104
 - values, 2105
- decaf::util::Map::Entry, 1570
 - ~Entry, 1570
 - Entry, 1570
 - getKey, 1570
 - getValue, 1570
 - setValue, 1570
- decaf::util::PriorityQueue, 2571
 - ~PriorityQueue, 2574
 - add, 2574
 - clear, 2574
 - comparator, 2574
 - iterator, 2575
 - offer, 2575
 - operator=, 2575, 2576
 - peek, 2576
 - poll, 2576
 - PriorityQueue, 2573, 2574
 - PriorityQueueIterator, 2578
 - remove, 2576, 2577
 - size, 2577

- decaf::util::Properties, 2657
 - ~Properties, 2659
 - clear, 2659
 - clone, 2659
 - copy, 2659
 - defaults, 2665
 - equals, 2659
 - getProperty, 2659, 2660
 - hasProperty, 2660
 - isEmpty, 2660
 - load, 2660, 2662
 - operator=, 2662
 - Properties, 2659
 - remove, 2663
 - setProperty, 2663
 - size, 2663
 - store, 2663, 2664
 - toArray, 2664
 - toString, 2665
- decaf::util::Queue, 2671
 - ~Queue, 2672
 - element, 2672
 - offer, 2672
 - peek, 2672
 - poll, 2673
 - remove, 2673
- decaf::util::Random, 2677
 - next, 2678
 - nextBoolean, 2679
 - nextBytes, 2679
 - nextDouble, 2679
 - nextFloat, 2679
 - nextGaussian, 2679
 - nextInt, 2680
 - nextLong, 2680
 - Random, 2678
 - setSeed, 2681
- decaf::util::Set, 2905
 - ~Set, 2905
- decaf::util::StlList, 3017
 - ~StlList, 3022
 - add, 3022
 - addAll, 3023
 - clear, 3023
 - contains, 3024
 - copy, 3024
 - equals, 3024
 - get, 3024
 - indexOf, 3025
 - isEmpty, 3025
 - iterator, 3025
 - lastIndexOf, 3026
 - listIterator, 3026, 3027
 - remove, 3027
 - set, 3028
 - size, 3028
 - StlList, 3021
- decaf::util::StlMap, 3030
 - ~StlMap, 3034
 - clear, 3034
 - containsKey, 3034
 - containsValue, 3035
 - copy, 3035
 - equals, 3035, 3036
 - get, 3036
 - isEmpty, 3037
 - keySet, 3037
 - lock, 3037
 - notify, 3037
 - notifyAll, 3038
 - put, 3038
 - putAll, 3038, 3039
 - remove, 3039
 - size, 3039
 - StlMap, 3033, 3034
 - tryLock, 3040
 - unlock, 3040
 - values, 3040
 - wait, 3040, 3041
- decaf::util::StlQueue, 3043
 - ~StlQueue, 3045
 - back, 3045
 - clear, 3045
 - empty, 3045
 - enqueueFront, 3046
 - front, 3046
 - getSafeValue, 3046
 - iterator, 3046
 - lock, 3046
 - notify, 3047
 - notifyAll, 3047
 - pop, 3047
 - push, 3047
 - reverse, 3048
 - size, 3048
 - StlQueue, 3045
 - toArray, 3048
 - tryLock, 3048
 - unlock, 3048
 - wait, 3049, 3050
- decaf::util::StlSet, 3051
 - ~StlSet, 3053
 - add, 3053
 - clear, 3054
 - contains, 3054
 - copy, 3055
 - equals, 3055
 - isEmpty, 3055

- iterator, 3055
- remove, 3055
- size, 3056
- StlSet, 3053
- decaf::util::StringTokenizer, 3094
 - ~StringTokenizer, 3095
 - countTokens, 3095
 - hasMoreTokens, 3095
 - nextToken, 3095
 - reset, 3096
 - StringTokenizer, 3094
 - toArray, 3096
- decaf::util::Timer, 3188
 - ~Timer, 3190
 - cancel, 3190
 - purge, 3190
 - schedule, 3190–3194
 - scheduleAtFixedRate, 3195–3197
 - Timer, 3190
- decaf::util::TimerTask, 3199
 - ~TimerTask, 3200
 - cancel, 3200
 - decaf::internal::util::TimerTaskHeap, 3201
 - getWhen, 3200
 - isScheduled, 3200
 - scheduledExecutionTime, 3200
 - setScheduledTime, 3200
 - Timer, 3201
 - TimerImpl, 3201
 - TimerTask, 3200
- decaf::util::UUID, 3356
 - ~UUID, 3358
 - clockSequence, 3358
 - compareTo, 3358
 - equals, 3358
 - fromString, 3358
 - getLeastSignificantBits, 3359
 - getMostSignificantBits, 3359
 - nameUUIDFromBytes, 3359
 - node, 3359
 - operator<, 3360
 - operator==, 3360
 - randomUUID, 3360
 - timestamp, 3360
 - toString, 3361
 - UUID, 3357
 - variant, 3361
 - version, 3361
- DECAF_API
 - decaf/util/Config.h, 3591
- DECAF_CATCH_EXCEPTION_CONVERT
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3536
- DECAF_CATCH_NOTHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3536
- DECAF_CATCH_RETHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3537
- DECAF_CATCHALL_NOTHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3537
- DECAF_CATCHALL_THROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3537
- DECAF_UNUSED
 - decaf/util/Config.h, 3591
- DecafRuntime
 - decaf::internal::DecafRuntime, 1472
- decode
 - decaf::internal::net::URLEncoderDecoder,
3319
 - decaf::lang::Byte, 843
 - decaf::lang::Integer, 1766
 - decaf::lang::Long, 2061
 - decaf::lang::Short, 2909
 - decaf::net::URLDecoder, 3349
- decreaseUsage
 - activemq::util::MemoryUsage, 2142
 - activemq::util::Usage, 3351
- decrementAndGet
 - decaf::util::concurrent::atomic::AtomicInteger,
635
- DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner,
1473
- DEFAULT_MAX_BLOCK_SIZE
 - decaf::util::concurrent::ThreadPool, 3180
- DEFAULT_MAX_POOL_SIZE
 - decaf::util::concurrent::ThreadPool, 3180
- DEFAULT_MESSAGE_SIZE
 - activemq::commands::Message, 2161
- DEFAULT_ORDERED_TARGET
 - activemq::commands::ActiveMQDestination,
283
- DEFAULT_PRIORITY
 - activemq::cmsutil::CmsTemplate, 1053
- DEFAULT_TIME_TO_LIVE
 - activemq::cmsutil::CmsTemplate, 1053
- DEFAULT_VERSION
 - activemq::wireformat::openwire::OpenWireFormat,
2459
- defaults
 - decaf::util::Properties, 2665
- deleteIfCancelled
 - decaf::internal::util::TimerTaskHeap, 3203
- deliverAcks
 - activemq::core::ActiveMQConsumer, 266

- activemq::core::ActiveMQSession, 453
- DELIVERY_MODE
 - cms::DeliveryMode, 1478
- deliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2256
- dequeue
 - activemq::core::ActiveMQConsumer, 266
 - activemq::core::MessageDispatchChannel, 2225
- dequeueNoWait
 - activemq::core::MessageDispatchChannel, 2226
- deQueueTask
 - decaf::util::concurrent::ThreadPool, 3177
- destination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2605
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2691
 - activemq::commands::ConsumerInfo, 1307
 - activemq::commands::DestinationInfo, 1488
 - activemq::commands::JournalQueueAck, 1837
 - activemq::commands::JournalTopicAck, 1862
 - activemq::commands::Message, 2161
 - activemq::commands::MessageAck, 2193
 - activemq::commands::MessageDispatch, 2223
 - activemq::commands::MessageDispatchNotification, 2256
 - activemq::commands::MessagePull, 2350
 - activemq::commands::ProducerInfo, 2635
 - activemq::commands::SubscriptionInfo, 3101
- DESTINATION_ADD_OPERATION
 - activemq::core::ActiveMQConstants, 261
- DESTINATION_REMOVE_OPERATION
 - activemq::core::ActiveMQConstants, 261
- DestinationActions
 - activemq::core::ActiveMQConstants, 261
- DestinationInfo
 - activemq::commands::DestinationInfo, 1485
- DestinationInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1502
 - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1490
 - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1494
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1498
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1506
- DestinationOption
 - activemq::core::ActiveMQConstants, 261
- DestinationType
 - cms::Destination, 1480
- destOptionMap
 - activemq::core::ActiveMQConstants::StaticInitializer, 3016
- destOptions
 - activemq::core::ActiveMQConstants::StaticInitializer, 3016
- destroy
 - activemq::cmsutil::CmsAccessor, 1026
 - activemq::cmsutil::CmsDestinationAccessor, 1029
 - activemq::cmsutil::CmsTemplate, 1044
 - activemq::cmsutil::DestinationResolver, 1509
 - activemq::cmsutil::DynamicDestinationResolver, 1568
 - activemq::cmsutil::ResourceLifecycleManager, 2779
 - activemq::commands::ActiveMQTempQueue, 525
 - activemq::commands::ActiveMQTempTopic, 550
 - cms::TemporaryQueue, 3166
 - cms::TemporaryTopic, 3168
 - decaf::internal::util::concurrent::ConditionImpl, 1126
 - decaf::internal::util::concurrent::MutexImpl, 2391
 - decaf::util::logging::LogManager, 2047
 - decaf::util::logging::LogWriter, 2055
- destroyDestination
 - activemq::core::ActiveMQConnection, 237
- destroyMarshalers
 - activemq::wireformat::openwire::OpenWireFormat, 2451
- digit
 - decaf::lang::Character, 984
- DISCONNECT
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- DiscoveryEvent
 - activemq::commands::DiscoveryEvent, 1512
- DiscoveryEventMarshaller
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1532

- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1516
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1520
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1524
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1528
- dispatch
 - activemq::core::ActiveMQConsumer, 267
 - activemq::core::ActiveMQSession, 453
 - activemq::core::Dispatcher, 1536
- dispatchAsync
 - activemq::commands::ConsumerInfo, 1307
 - activemq::commands::ProducerInfo, 2635
- DispatchData
 - activemq::core::DispatchData, 1535
- disposeOf
 - activemq::core::ActiveMQSession, 453
- doClose
 - activemq::core::ActiveMQConsumer, 267
- doCreateComposite
 - activemq::transport::failover::FailoverTransportFactory, 1625
 - activemq::transport::mock::MockTransportFactory, 2383
 - activemq::transport::tcp::TcpTransportFactory, 3164
- doInCms
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2604
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2690
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2836
 - activemq::cmsutil::ProducerCallback, 2603
 - activemq::cmsutil::SessionCallback, 2852
- doStartTransaction
 - activemq::core::ActiveMQSession, 454
- Double
 - decaf::lang::Double, 1539
- DOUBLE_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- DoubleArrayBuffer
 - decaf::internal::nio::DoubleArrayBuffer, 1549, 1550
- DoubleBuffer
 - decaf::nio::DoubleBuffer, 1558
- doubleToLongBits
 - decaf::lang::Double, 1540
- doubleToRawLongBits
 - decaf::lang::Double, 1541
- doubleValue
 - decaf::lang::Double, 1541
 - decaf::lang::Character, 985
 - decaf::lang::Float, 1650
 - decaf::lang::Long, 2061
 - decaf::lang::Number, 2433
 - decaf::lang::Short, 2909
 - decaf::util::concurrent::atomic::AtomicInteger, 635
- doUnmarshal
 - activemq::wireformat::openwire::OpenWireFormat, 2451
- drainPermits
 - decaf::util::concurrent::Semaphore, 2832
- drainTo
 - decaf::util::concurrent::SynchronousQueue, 3140
- droppable
 - activemq::commands::Message, 2161
- duplexConnection
 - activemq::commands::BrokerInfo, 782
- duplicate
 - decaf::internal::nio::ByteBuffer, 879
 - decaf::internal::nio::CharArrayBuffer, 994
 - decaf::internal::nio::DoubleArrayBuffer, 1552
 - decaf::internal::nio::FloatArrayBuffer, 1662
 - decaf::internal::nio::IntArrayBuffer, 1748
 - decaf::internal::nio::LongArrayBuffer, 2075
 - decaf::internal::nio::ShortArrayBuffer, 2919
 - decaf::nio::ByteBuffer, 922
 - decaf::nio::CharBuffer, 1005
 - decaf::nio::DoubleBuffer, 1560
 - decaf::nio::FloatBuffer, 1669
 - decaf::nio::IntBuffer, 1755
 - decaf::nio::LongBuffer, 2083
 - decaf::nio::ShortBuffer, 2927
- DUPS_OK_ACKNOWLEDGE
 - cms::Session, 2842
- dynamicCast
 - decaf::lang::Pointer, 2500
- E
 - decaf::lang::Math, 2140
- element
 - decaf::util::AbstractQueue, 164
 - decaf::util::Queue, 2672
- empty
 - decaf::util::StlQueue, 3045
- encode
 - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 2552
 - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 2552
 - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 2552
 - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 2552
- activemq::util::PrimitiveValueNode::PrimitiveValue, 2552

- decaf::net::URLEncoder, 3350
- encodeOthers
 - decaf::internal::net::URLEncoderDecoder, 3320
- enqueue
 - activemq::core::MessageDispatchChannel, 2226
- enqueueFirst
 - activemq::core::MessageDispatchChannel, 2226
- enqueueFront
 - decaf::util::StlQueue, 3046
- enqueueUsage
 - activemq::util::MemoryUsage, 2142
 - activemq::util::Usage, 3351
- Entry
 - decaf::util::Map::Entry, 1570
- entry
 - decaf::util::logging::Logger, 2030
- EOFException
 - decaf::io::EOFException, 1571, 1572
- equals
 - activemq::commands::ActiveMQBlobMessage, 174
 - activemq::commands::ActiveMQBytesMessage, 201
 - activemq::commands::ActiveMQDestination, 277
 - activemq::commands::ActiveMQMapMessage, 312
 - activemq::commands::ActiveMQMessage, 343
 - activemq::commands::ActiveMQMessageTemplate, 369
 - activemq::commands::ActiveMQObjectMessage, 384
 - activemq::commands::ActiveMQQueue, 421
 - activemq::commands::ActiveMQStreamMessage, 468
 - activemq::commands::ActiveMQTempDestination, 500
 - activemq::commands::ActiveMQTempQueue, 525
 - activemq::commands::ActiveMQTempTopic, 550
 - activemq::commands::ActiveMQTextMessage, 575
 - activemq::commands::ActiveMQTopic, 600
 - activemq::commands::BaseCommand, 651
 - activemq::commands::BaseDataStructure, 717
 - activemq::commands::BooleanExpression, 738
 - activemq::commands::BrokerId, 752
 - activemq::commands::BrokerInfo, 777
 - activemq::commands::ConnectionControl, 1136
 - activemq::commands::ConnectionError, 1161
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConnectionInfo, 1213
 - activemq::commands::ConsumerControl, 1251
 - activemq::commands::ConsumerId, 1277
 - activemq::commands::ConsumerInfo, 1302
 - activemq::commands::ControlCommand, 1331
 - activemq::commands::DataArrayResponse, 1357
 - activemq::commands::DataResponse, 1396
 - activemq::commands::DataStructure, 1463
 - activemq::commands::DestinationInfo, 1485
 - activemq::commands::DiscoveryEvent, 1512
 - activemq::commands::ExceptionResponse, 1583
 - activemq::commands::FlushCommand, 1677
 - activemq::commands::IntegerResponse, 1778
 - activemq::commands::JournalQueueAck, 1835
 - activemq::commands::JournalTopicAck, 1859
 - activemq::commands::JournalTrace, 1884
 - activemq::commands::JournalTransaction, 1907
 - activemq::commands::KeepAliveInfo, 1931
 - activemq::commands::LastPartialCommand, 1959
 - activemq::commands::LocalTransactionId, 1994, 1995
 - activemq::commands::Message, 2150
 - activemq::commands::MessageAck, 2189
 - activemq::commands::MessageDispatch, 2220
 - activemq::commands::MessageDispatchNotification, 2252
 - activemq::commands::MessageId, 2280, 2281
 - activemq::commands::MessagePull, 2347
 - activemq::commands::NetworkBridgeFilter, 2394
 - activemq::commands::PartialCommand, 2474

- activemq::commands::ProducerAck, 2580
- activemq::commands::ProducerId, 2608
- activemq::commands::ProducerInfo, 2632
- activemq::commands::RemoveInfo, 2704
- activemq::commands::RemoveSubscriptionInfo, 2728
- activemq::commands::ReplayCommand, 2753
- activemq::commands::Response, 2782
- activemq::commands::SessionId, 2855
- activemq::commands::SessionInfo, 2878
- activemq::commands::ShutdownInfo, 2935
- activemq::commands::SubscriptionInfo, 3098
- activemq::commands::TransactionId, 3218
- activemq::commands::TransactionInfo, 3241
- activemq::commands::WireFormatInfo, 3372
- activemq::commands::XATransactionId, 3412
- decaf::lang::Boolean, 734
- decaf::lang::Byte, 843
- decaf::lang::Character, 985
- decaf::lang::Comparable, 1084
- decaf::lang::Double, 1541, 1542
- decaf::lang::Float, 1651
- decaf::lang::Integer, 1766, 1767
- decaf::lang::Long, 2061, 2062
- decaf::lang::Short, 2909
- decaf::net::URI, 3312
- decaf::nio::ByteBuffer, 922
- decaf::nio::CharBuffer, 1005
- decaf::nio::DoubleBuffer, 1560
- decaf::nio::FloatBuffer, 1669
- decaf::nio::IntBuffer, 1755
- decaf::nio::LongBuffer, 2083
- decaf::nio::ShortBuffer, 2927
- decaf::security::cert::Certificate, 969
- decaf::security::Principal, 2569
- decaf::security_-
 provider::unix::openssl::OpenSSLX500Principal, 2440
- decaf::security_-
 provider::unix::openssl::OpenSSLX509Certificate, 2443
- decaf::util::AbstractCollection, 153
- decaf::util::Collection, 1059
- decaf::util::concurrent::ConcurrentStlMap, 1108
- decaf::util::concurrent::SynchronousQueue, 3140
- decaf::util::concurrent::TimeUnit, 3208
- decaf::util::Date, 1469
- decaf::util::Map, 2098
- decaf::util::Properties, 2659
- decaf::util::StlList, 3024
- decaf::util::StlMap, 3035, 3036
- decaf::util::StlSet, 3055
- decaf::util::UUID, 3358
- Error
 decaf::util::logging, 144
- error
 decaf::util::logging::Logger, 2030
- decaf::util::logging::SimpleLogger, 2963
- ERROR_CMD
 activemq::wireformat::stomp::StompCommandConstants, 3059
- Exception
 decaf::lang::Exception, 1575, 1576
- exception
 activemq::commands::ConnectionError, 1163
- activemq::commands::ExceptionResponse, 1584
- ExceptionResponse
 activemq::commands::ExceptionResponse, 1583
- ExceptionResponseMarshaller
 activemq::wireformat::openwire::marshal::v1::ExceptionResponse, 1590
- activemq::wireformat::openwire::marshal::v2::ExceptionResponse, 1586
- activemq::wireformat::openwire::marshal::v3::ExceptionResponse, 1594
- activemq::wireformat::openwire::marshal::v4::ExceptionResponse, 1598
- activemq::wireformat::openwire::marshal::v5::ExceptionResponse, 1602
- exclusive
 activemq::commands::ActiveMQDestination, 283
- activemq::commands::ConsumerInfo, 1307
- execute
 activemq::cmsutil::CmsTemplate, 1044
- activemq::core::ActiveMQSessionExecutor, 461
- decaf::util::concurrent::Executor, 1609
- executeFirst
 activemq::core::ActiveMQSessionExecutor, 461
- ExecutionException
 decaf::util::concurrent::ExecutionException, 1605, 1606
- exit
 activemq::commands::ConnectionControl, 1139

- decaf::util::logging::Logger, 2031
- expiration
 - activemq::commands::Message, 2161
- failIfReadOnlyBody
 - activemq::commands::ActiveMQMessageTemplate, 369
- failIfReadOnlyProperties
 - activemq::commands::ActiveMQMessageTemplate, 369
- failIfWriteOnlyBody
 - activemq::commands::ActiveMQMessageTemplate, 369
- FailoverTransport
 - activemq::transport::failover::FailoverTransport, 1614
- FailoverTransportListener
 - activemq::transport::failover::FailoverTransport, 1623
 - activemq::transport::failover::FailoverTransportListener, 1628
- Fatal
 - decaf::util::logging, 144
- fatal
 - decaf::util::logging::Logger, 2031
 - decaf::util::logging::SimpleLogger, 2963
- faultTolerant
 - activemq::commands::ConnectionControl, 1139
- faultTolerantConfiguration
 - activemq::commands::BrokerInfo, 782
- FileName
 - activemq::commands::BrokerError::StackTraceElement, 2993
- FilterInputStream
 - decaf::io::FilterInputStream, 1633
- FilterOutputStream
 - decaf::io::FilterOutputStream, 1641
- find
 - decaf::internal::util::TimerTaskHeap, 3203
- findFactory
 - activemq::transport::TransportRegistry, 3292
 - activemq::wireformat::WireFormatRegistry, 3401
- fire
 - activemq::core::ActiveMQConnection, 238
 - activemq::core::ActiveMQSession, 454
 - activemq::transport::TransportFilter, 3283
- fireCommand
 - activemq::transport::mock::MockTransport, 2373
- fireException
 - activemq::transport::mock::MockTransport, 2374
- firstMessageId
 - activemq::commands::MessageAck, 2193
- firstNakNumber
 - activemq::commands::ReplayCommand, 2755
- flip
 - decaf::nio::Buffer, 806
- Float
 - decaf::lang::Float, 1649
- FLOAT_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1659, 1660
- FloatBuffer
 - decaf::nio::FloatBuffer, 1667
- floatToIntBits
 - decaf::lang::Float, 1651
- floatToRawIntBits
 - decaf::lang::Float, 1651
- floatValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2552
- decaf::lang::Byte, 844
- decaf::lang::Character, 985
- decaf::lang::Double, 1542
- decaf::lang::Float, 1652
- decaf::lang::Integer, 1767
- decaf::lang::Long, 2062
- decaf::lang::Number, 2433
- decaf::lang::Short, 2909
- decaf::util::concurrent::atomic::AtomicInteger, 635
- floor
 - decaf::lang::Math, 2130
- flush
 - activemq::commands::ConsumerControl, 1254
 - decaf::internal::io::StandardErrorOutputStream, 2995
 - decaf::internal::io::StandardOutputStream, 3009
 - decaf::io::BufferedOutputStream, 815
 - decaf::io::ByteArrayOutputStream, 902
 - decaf::io::FilterOutputStream, 1642
 - decaf::io::OutputStream, 2470
 - decaf::net::SocketOutputStream, 2985
 - decaf::util::logging::Handler, 1710
 - decaf::util::logging::StreamHandler, 3078
- FlushCommand
 - activemq::commands::FlushCommand, 1677

- FlushCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1669–1671
 - 1684
 - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 2084, 2085
 - 1680
 - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 2928, 2929
 - 1688
 - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 631
 - 1692
 - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 635
 - 1696
- format
 - decaf::util::logging::Formatter, 1699
 - decaf::util::logging::SimpleFormatter, 2960
- formatId
 - activemq::commands::XATransactionId, 3414
- formatMessage
 - decaf::util::logging::Formatter, 1699
 - decaf::util::logging::SimpleFormatter, 2960
- fromStream
 - activemq::wireformat::stomp::StompFrame, 3062
- fromString
 - decaf::util::UUID, 3358
- front
 - decaf::util::StlQueue, 3046
- FutureResponse
 - activemq::transport::correlator::FutureResponse, 1704
- GeneralSecurityException
 - decaf::security::GeneralSecurityException, 1706, 1707
- generation
 - decaf::util::concurrent::ConditionHandle, 1124
- get
 - decaf::internal::nio::ByteBuffer, 879, 880
 - decaf::internal::nio::CharArrayBuffer, 994, 995
 - decaf::internal::nio::DoubleArrayBuffer, 1552
 - decaf::internal::nio::FloatArrayBuffer, 1662
 - decaf::internal::nio::IntArrayBuffer, 1748
 - decaf::internal::nio::LongArrayBuffer, 2075
 - decaf::internal::nio::ShortArrayBuffer, 2919
 - decaf::internal::util::ByteArrayAdapter, 856
 - decaf::lang::Pointer, 2500
 - decaf::nio::ByteBuffer, 922, 923
 - decaf::nio::CharBuffer, 1005, 1006
 - decaf::nio::DoubleBuffer, 1561, 1562
 - decaf::nio::FloatBuffer, 1669–1671
 - decaf::nio::IntBuffer, 1755–1757
 - decaf::nio::LongBuffer, 2084, 2085
 - decaf::nio::ShortBuffer, 2928, 2929
 - decaf::util::concurrent::atomic::AtomicBoolean, 631
 - decaf::util::concurrent::atomic::AtomicInteger, 635
 - decaf::util::concurrent::atomic::AtomicReference, 642
 - decaf::util::concurrent::ConcurrentStlMap, 1108, 1109
 - decaf::util::concurrent::Future, 1702
 - decaf::util::List, 1986
 - decaf::util::Map, 2098, 2099
 - decaf::util::StlList, 3024
 - decaf::util::StlMap, 3036
- getAckHandler
 - activemq::commands::Message, 2150
- getAckMode
 - activemq::commands::SessionInfo, 2878
- getAcknowledgeMode
 - activemq::cmsutil::PooledSession, 2515
 - activemq::core::ActiveMQSession, 454
 - cms::Session, 2849
- getAckType
 - activemq::commands::MessageAck, 2190
- getAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1303
- getAlgorithm
 - decaf::security::Key, 1954
- getAndAdd
 - decaf::util::concurrent::atomic::AtomicInteger, 636
- getAndDecrement
 - decaf::util::concurrent::atomic::AtomicInteger, 636
- getAndIncrement
 - decaf::util::concurrent::atomic::AtomicInteger, 636
- getAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 631
 - decaf::util::concurrent::atomic::AtomicInteger, 636
 - decaf::util::concurrent::atomic::AtomicReference, 642
- getAnonymousLogger
 - decaf::util::logging::Logger, 2031
- getAprPool
 - decaf::internal::AprPool, 628
- getArrival
 - activemq::commands::Message, 2151

- getAuthority
 - decaf::internal::net::URIType, 3341
 - decaf::net::URI, 3312
- getBacklog
 - decaf::util::concurrent::ThreadPool, 3177
- getBackOffMultiplier
 - activemq::transport::failover::FailoverTransport, 1512, 1513
- getBackup
 - activemq::transport::failover::BackupTransportPool, 648
- getBackupPoolSize
 - activemq::transport::failover::BackupTransportPool, 648
 - activemq::transport::failover::FailoverTransport, 1616
- getBasicConstraints
 - decaf::security::cert::X509Certificate, 3408
 - decaf::security_ - provider::unix::openssl::OpenSSLX509Certificate, 2443
- getBlockSize
 - decaf::util::concurrent::ThreadPool, 3177
- getBody
 - activemq::wireformat::stomp::StompFrame, 3063
- getBodyBytes
 - activemq::commands::ActiveMQBytesMessage, 201
 - cms::BytesMessage, 941
- getBodyLength
 - activemq::commands::ActiveMQBytesMessage, 201
 - activemq::wireformat::stomp::StompFrame, 3063
 - cms::BytesMessage, 942
- getBool
 - activemq::util::PrimitiveList, 2528
 - activemq::util::PrimitiveMap, 2539
 - activemq::util::PrimitiveValueNode, 2561
- getBoolean
 - activemq::commands::ActiveMQMapMessage, 312
 - cms::MapMessage, 2108
- getBooleanProperty
 - activemq::commands::ActiveMQMessageTemplate, 369
 - activemq::wireformat::openwire::utils::MessagePropertyInterface, 2341
 - cms::Message, 2169
- getBranchQualifier
 - activemq::commands::XATransactionId, 3412
- getBrokerId
 - activemq::commands::BrokerInfo, 777, 778
- getBrokerInTime
 - activemq::commands::Message, 2152
- getBrokerName
 - activemq::commands::BrokerInfo, 778
 - activemq::commands::DiscoveryEvent, 1512, 1513
- getBrokerOutTime
 - activemq::commands::Message, 2152
- getBrokerPath
 - activemq::commands::ConnectionInfo, 1213
- getBrokerSequenceId
 - activemq::commands::ConsumerInfo, 1303
 - activemq::commands::DestinationInfo, 1486
 - activemq::commands::Message, 2152
 - activemq::commands::ProducerInfo, 2633
- getBrokerUploadUrl
 - activemq::commands::MessageId, 2281
- getBrokerUploadUrl
 - activemq::commands::BrokerInfo, 778
- getBrokerURL
 - activemq::commands::BrokerInfo, 778
 - activemq::core::ActiveMQConnectionFactory, 247
- getByte
 - activemq::commands::ActiveMQMapMessage, 312
 - activemq::util::PrimitiveList, 2528
 - activemq::util::PrimitiveMap, 2539
 - activemq::util::PrimitiveValueNode, 2562
 - cms::MapMessage, 2108
- getByteArray
 - activemq::util::PrimitiveList, 2528
 - activemq::util::PrimitiveMap, 2539
 - activemq::util::PrimitiveValueNode, 2562
 - decaf::internal::util::ByteArrayAdapter, 856
- getByteProperty
 - activemq::commands::ActiveMQMessageTemplate, 370
 - activemq::wireformat::openwire::utils::MessagePropertyInterface, 2341
 - cms::Message, 2169
- getBytes
 - activemq::commands::ActiveMQMapMessage, 313
 - cms::MapMessage, 2109
- getCacheSize
 - activemq::commands::WireFormatInfo, 3372
 - activemq::wireformat::openwire::OpenWireFormat, 2452
- getCapacity
 -

- decaf::internal::util::ByteArrayAdapter, 856
- getCause
 - activemq::commands::BrokerError, 746
 - cms::CMSException, 1032
 - decaf::lang::Exception, 1577
 - decaf::lang::Throwable, 3182
- getChar
 - activemq::commands::ActiveMQMapMessage, 313
 - activemq::util::PrimitiveList, 2529
 - activemq::util::PrimitiveMap, 2540
 - activemq::util::PrimitiveValueNode, 2562
 - cms::MapMessage, 2109
 - decaf::internal::nio::ByteBuffer, 880
 - decaf::internal::util::ByteArrayAdapter, 856
 - decaf::nio::ByteBuffer, 924
- getCharArray
 - decaf::internal::util::ByteArrayAdapter, 857
- getCharCapacity
 - decaf::internal::util::ByteArrayAdapter, 857
- getClientID
 - activemq::core::ActiveMQConnection, 238
 - cms::Connection, 1133
- getClientId
 - activemq::commands::ActiveMQDestination, 278
 - activemq::commands::ConnectionInfo, 1213
 - activemq::commands::JournalTopicAck, 1860
 - activemq::commands::RemoveSubscriptionInfo, 2729
 - activemq::commands::SubscriptionInfo, 3099
 - activemq::core::ActiveMQConnectionSupport, 255
- getCloseTimeout
 - activemq::core::ActiveMQConnectionSupport, 255
- getCluster
 - activemq::commands::Message, 2152
- getCMSCorrelationID
 - activemq::commands::ActiveMQMessageTemplate, 370
 - cms::Message, 2169
- getCMSDeliveryMode
 - activemq::commands::ActiveMQMessageTemplate, 370
 - cms::Message, 2170
- getCMSDestination
 - activemq::commands::ActiveMQDestination, 278
 - activemq::commands::ActiveMQMessageTemplate, 371
 - activemq::commands::ActiveMQQueue, 421
 - activemq::commands::ActiveMQTempQueue, 525
 - activemq::commands::ActiveMQTempTopic, 550
 - activemq::commands::ActiveMQTopic, 600
 - cms::Message, 2170
- getCMSExpiration
 - activemq::commands::ActiveMQMessageTemplate, 371
 - cms::Message, 2171
- getCMSMajorVersion
 - activemq::core::ActiveMQConnectionMetaData, 250
 - cms::ConnectionMetaData, 1238
- getCMSMessageID
 - activemq::commands::ActiveMQMessageTemplate, 371
 - cms::Message, 2171
- getCMSMinorVersion
 - activemq::core::ActiveMQConnectionMetaData, 250
 - cms::ConnectionMetaData, 1238
- getCMSPriority
 - activemq::commands::ActiveMQMessageTemplate, 371
 - cms::Message, 2172
- getCMSProperties
 - activemq::commands::ActiveMQQueue, 421
 - activemq::commands::ActiveMQTempQueue, 525
 - activemq::commands::ActiveMQTempTopic, 550
 - activemq::commands::ActiveMQTopic, 600
 - cms::Destination, 1481
- getCMSProviderName
 - activemq::core::ActiveMQConnectionMetaData, 250
 - cms::ConnectionMetaData, 1238
- getCMSRedelivered
 - activemq::commands::ActiveMQMessageTemplate, 372
 - cms::Message, 2172
- getCMSReplyTo
 - activemq::commands::ActiveMQMessageTemplate, 372
 - cms::Message, 2173
- getCMSTimeStamp
 - activemq::commands::ActiveMQMessageTemplate, 371
 - cms::Message, 2171

- activemq::commands::ActiveMQMessageTemplate
 - 372
- cms::Message, 2173
- getCMSType
 - activemq::commands::ActiveMQMessageTemplate
 - 373
 - cms::Message, 2174
- getCMSVersion
 - activemq::core::ActiveMQConnectionMetaData,
 - 251
 - cms::ConnectionMetaData, 1238
- getCMSXPathPropertyNames
 - activemq::core::ActiveMQConnectionMetaData,
 - 251
 - cms::ConnectionMetaData, 1239
- getCommand
 - activemq::commands::ControlCommand,
 - 1331, 1332
 - activemq::wireformat::stomp::StompFrame,
 - 3063
- getCommandId
 - activemq::commands::BaseCommand, 652
 - activemq::commands::Command, 1064
 - activemq::commands::PartialCommand,
 - 2474
- getCommands
 - activemq::state::TransactionState, 3266
- getComponents
 - activemq::util::CompositeData, 1089
- getConnection
 - activemq::core::ActiveMQSession, 454
- getConnectionFactory
 - activemq::cmsutil::CmsAccessor, 1026
- getConnectionId
 - activemq::commands::BrokerInfo, 778
 - activemq::commands::ConnectionError,
 - 1161, 1162
 - activemq::commands::ConnectionInfo,
 - 1213
 - activemq::commands::ConsumerId, 1277
 - activemq::commands::DestinationInfo,
 - 1486
 - activemq::commands::LocalTransactionId,
 - 1995
 - activemq::commands::ProducerId, 2608
 - activemq::commands::RemoveSubscriptionInfo,
 - 2729
 - activemq::commands::SessionId, 2855
 - activemq::commands::TransactionInfo,
 - 3241, 3242
 - activemq::core::ActiveMQConnection, 238
- getConnectionInfo
 - activemq::core::ActiveMQConnection, 238
- getConsumerId
 - activemq::commands::ActiveMQMessageTemplate
 - 372
 - activemq::commands::ConsumerControl,
 - 1252
 - activemq::commands::ConsumerInfo, 1303
 - activemq::commands::MessageAck, 2190
 - activemq::commands::MessageDispatch,
 - 2221
 - activemq::commands::MessageDispatchNotification,
 - 2253
 - activemq::commands::MessagePull, 2348
 - activemq::core::ActiveMQConsumer, 267
 - activemq::core::DispatchData, 1535
- getConsumerInfo
 - activemq::core::ActiveMQConsumer, 267
- getConsumerState
 - activemq::state::SessionState, 2904
- getConsumerStates
 - activemq::state::SessionState, 2904
- getContent
 - activemq::commands::Message, 2152
- getCorrelationId
 - activemq::commands::Message, 2152
 - activemq::commands::MessagePull, 2348
 - activemq::commands::Response, 2783
- getCount
 - decaf::util::concurrent::CountDownLatch,
 - 1355
- getData
 - activemq::commands::DataArrayResponse,
 - 1357, 1358
 - activemq::commands::DataResponse, 1396,
 - 1397
 - activemq::commands::PartialCommand,
 - 2475
- getDataStructure
 - activemq::commands::Message, 2152
- getDataStructureType
 - activemq::commands::ActiveMQBlobMessage,
 - 174
 - activemq::commands::ActiveMQBytesMessage,
 - 202
 - activemq::commands::ActiveMQDestination,
 - 278
 - activemq::commands::ActiveMQMapMessage,
 - 313
 - activemq::commands::ActiveMQMessage,
 - 343
 - activemq::commands::ActiveMQObjectMessage,
 - 385
 - activemq::commands::ActiveMQQueue,
 - 421
 - activemq::commands::ActiveMQStreamMessage,
 - 468
 - activemq::commands::ActiveMQTempDestination,
 - 501

- activemq::commands::ActiveMQTempQueue, 526
- activemq::commands::ActiveMQTempTopic, 551
- activemq::commands::ActiveMQTextMessage, 575
- activemq::commands::ActiveMQTopic, 600
- activemq::commands::BrokerError, 747
- activemq::commands::BrokerId, 753
- activemq::commands::BrokerInfo, 778
- activemq::commands::ConnectionControl, 1137
- activemq::commands::ConnectionError, 1162
- activemq::commands::ConnectionId, 1189
- activemq::commands::ConnectionInfo, 1213
- activemq::commands::ConsumerControl, 1252
- activemq::commands::ConsumerId, 1277
- activemq::commands::ConsumerInfo, 1303
- activemq::commands::ControlCommand, 1332
- activemq::commands::DataArrayResponse, 1358
- activemq::commands::DataResponse, 1397
- activemq::commands::DataStructure, 1464
- activemq::commands::DestinationInfo, 1486
- activemq::commands::DiscoveryEvent, 1513
- activemq::commands::ExceptionResponse, 1583
- activemq::commands::FlushCommand, 1677
- activemq::commands::IntegerResponse, 1778
- activemq::commands::JournalQueueAck, 1835
- activemq::commands::JournalTopicAck, 1860
- activemq::commands::JournalTrace, 1884
- activemq::commands::JournalTransaction, 1907
- activemq::commands::KeepAliveInfo, 1931
- activemq::commands::LastPartialCommand, 1959
- activemq::commands::LocalTransactionId, 1995
- activemq::commands::Message, 2152
- activemq::commands::MessageAck, 2190
- activemq::commands::MessageDispatch, 2221
- activemq::commands::MessageDispatchNotification, 2253
- activemq::commands::MessageId, 2281
- activemq::commands::MessagePull, 2348
- activemq::commands::NetworkBridgeFilter, 2394
- activemq::commands::PartialCommand, 2475
- activemq::commands::ProducerAck, 2580
- activemq::commands::ProducerId, 2608
- activemq::commands::ProducerInfo, 2633
- activemq::commands::RemoveInfo, 2704
- activemq::commands::RemoveSubscriptionInfo, 2729
- activemq::commands::ReplayCommand, 2753
- activemq::commands::Response, 2783
- activemq::commands::SessionId, 2855
- activemq::commands::SessionInfo, 2879
- activemq::commands::ShutdownInfo, 2935
- activemq::commands::SubscriptionInfo, 3099
- activemq::commands::TransactionId, 3219
- activemq::commands::TransactionInfo, 3242
- activemq::commands::WireFormatInfo, 3373
- activemq::commands::XATransactionId, 3412
- activemq::wireformat::openwire::marshal::DataStreamMarshal, 1424
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 182
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesL, 218
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapM, 327
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessa, 350
- activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 391
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 428
- activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 484
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 533
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemp7, 558
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextM, 583
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicL, 607

activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	760	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2301
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	788	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2360
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	1145	activemq::wireformat::openwire::marshal::v1::NetworkBridgeFormat	2410
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	1169	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2490
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	1196	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	2600
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1226	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2624
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	1264	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2649
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	1289	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	2708
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1314	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	2741
activemq::wireformat::openwire::marshal::v1::ConsumerQueueMarshaller	1343	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	2773
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	1368	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	2807
activemq::wireformat::openwire::marshal::v1::DataResponseV2Marshaller	1403	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	2862
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1502	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	2882
activemq::wireformat::openwire::marshal::v1::DiscoveryInfoMarshaller	1532	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	2938
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1590	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3103
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	1684	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3246
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	1785	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3396
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	1851	activemq::wireformat::openwire::marshal::v1::XATransactionInfoMarshaller	3420
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	1868	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	194
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	1899	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	230
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoV2Marshaller	1923	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	339
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	1950	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	362
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	1966	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	403
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	2010	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	440
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2211	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	496
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2244	activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	545
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2270	activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	570

activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2232
595	
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMessageMarshaller	2258
619	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2285
772	
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2352
800	
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2398
1157	
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2478
1181	
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2584
1208	
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2612
1234	
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2637
1272	
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2712
1297	
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	2737
1326	
activemq::wireformat::openwire::marshal::v2::ContactCommandMarshaller	2757
1351	
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	2792
1376	
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2866
1415	
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	2890
1490	
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	2950
1516	
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	3119
1586	
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3262
1680	
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	3388
1781	
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3428
1839	
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	178
1864	
activemq::wireformat::openwire::marshal::v2::JournalTopicInfoMarshaller	214
1887	
activemq::wireformat::openwire::marshal::v2::JournalTopicInfoMarshaller	323
1911	
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	346
1934	
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	387
1962	
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	424
1998	
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	480
2195	

activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	529	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	2002
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	554	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2203
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	579	activemq::wireformat::openwire::marshal::v3::MessageDispatchInfoMarshaller	2240
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMessageMarshaller	603	activemq::wireformat::openwire::marshal::v3::MessageDispatchInfoMarshaller	2266
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	756	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2293
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	784	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2356
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1141	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFormatMarshaller	2402
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1165	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2482
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1192	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	2588
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1218	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	2616
activemq::wireformat::openwire::marshal::v3::ConsumerGroupWithMarshaller	1256	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	2653
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1281	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	2716
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1310	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	2745
activemq::wireformat::openwire::marshal::v3::ContactCommandMarshaller	1335	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	2761
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	1360	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	2797
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1399	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	2870
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1494	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	2898
activemq::wireformat::openwire::marshal::v3::DiscardResponseMarshaller	1520	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	2946
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1594	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3111
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	1688	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3254
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	1789	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	3380
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	1847	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	3416
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	1872	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller	186
activemq::wireformat::openwire::marshal::v3::JournalFrameMarshaller	1895	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller	222
activemq::wireformat::openwire::marshal::v3::JournalFrameActiveMarshaller	1915	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller	331
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	1942	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	354
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	1970	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller	395

activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	1946	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	1946
432		activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	1974
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1974	activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller	2006
488		activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2207
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatMarshaller	2006	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2236
537		activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2262
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatMarshaller	2207	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2289
562		activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2368
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatMarshaller	2236	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFormatMarshaller	2414
587		activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2486
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatMarshaller	2262	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2592
611		activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2628
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2289	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2645
764		activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	2720
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2368	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller	2749
792		activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	2769
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2414	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	2812
1149		activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	2874
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2486	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	2886
1173		activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	2942
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	2592	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	3107
1200		activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3250
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	2628	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	3392
1222		activemq::wireformat::openwire::marshal::v4::XATransactionInfoMarshaller	3424
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller	2645	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMarshaller	190
1260		activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMarshaller	226
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	2720	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMarshaller	335
1285			
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	2749		
1318			
activemq::wireformat::openwire::marshal::v4::ContainerCommandMarshaller	2769		
1339			
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	2812		
1364			
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	2874		
1411			
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	2886		
1498			
activemq::wireformat::openwire::marshal::v4::DiscardResponseMarshaller	2942		
1524			
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	3107		
1598			
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	3250		
1692			
activemq::wireformat::openwire::marshal::v4::IntegratedResponseMarshaller	3392		
1793			
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	3424		
1843			
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	190		
1876			
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	226		
1891			
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	335		
1919			

activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	1903	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	
358		activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	1927	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	
399		436	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	1938	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	
436		492	
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaller	1978	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
492		541	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	2014	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller	
541		566	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	2199	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	
566		591	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2248	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	
591		615	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2274	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	
615		768	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2297	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	
768		796	
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2364	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	
796		1153	
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2406	activemq::wireformat::openwire::marshal::v5::NetworkBridgeFactoryMarshaller	
1153		1177	
activemq::wireformat::openwire::marshal::v5::ConnectionFromMarshaller	2494	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	
1177		1204	
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	2596	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	
1204		1230	
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	2620	activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	
1230		1268	
activemq::wireformat::openwire::marshal::v5::ConsumerGroupMarshaller	2641	activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	
1268		1293	
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	2724	activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	
1293		1322	
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	2733	activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionMarshaller	
1322		1347	
activemq::wireformat::openwire::marshal::v5::ContainerCommandMarshaller	2765	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller	
1347		1372	
activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller	2802	activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	
1372		1407	
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	2858	activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	
1407		1506	
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	2894	activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	
1506		1528	
activemq::wireformat::openwire::marshal::v5::DiscoveryInfoMarshaller	2954	activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	
1528		1602	
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	3115	activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	
1602		1696	
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	3258	activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	
1696		1797	
activemq::wireformat::openwire::marshal::v5::IntegrationResponseMarshaller	3384	activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	
1797		1855	
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	3432	activemq::wireformat::openwire::marshal::v5::XATransactionInfoMarshaller	
1855		1880	
activemq::wireformat::openwire::marshal::v5::JetDataInfoMarshaller,		activemq::cmsutil::CmsTemplate,	1045,

- 1046
- getDefaultDestinationName
 - activemq::cmsutil::CmsTemplate, 1046
- getDelay
 - decaf::util::concurrent::Delayed, 1477
- getDeliveryMode
 - activemq::cmsutil::CachedProducer, 958
 - activemq::cmsutil::CmsTemplate, 1046
 - activemq::core::ActiveMQProducer, 408
 - cms::MessageProducer, 2333
- getDeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2253
- getDestination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2604
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2691
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2776
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2777
 - activemq::commands::ConsumerInfo, 1303, 1304
 - activemq::commands::DestinationInfo, 1486, 1487
 - activemq::commands::JournalQueueAck, 1836
 - activemq::commands::JournalTopicAck, 1860, 1861
 - activemq::commands::Message, 2153
 - activemq::commands::MessageAck, 2190, 2191
 - activemq::commands::MessageDispatch, 2221, 2222
 - activemq::commands::MessageDispatchNotification, 2254
 - activemq::commands::MessagePull, 2348, 2349
 - activemq::commands::ProducerInfo, 2633, 2634
 - activemq::commands::SubscriptionInfo, 3099, 3100
- getDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 1029
- getDestinationType
 - activemq::commands::ActiveMQDestination, 279
 - activemq::commands::ActiveMQQueue, 421
 - activemq::commands::ActiveMQTempQueue, 526
- activemq::commands::ActiveMQTempTopic, 551
- activemq::commands::ActiveMQTopic, 600
- cms::Destination, 1481
- getDisableMessageID
 - activemq::cmsutil::CachedProducer, 958
 - activemq::core::ActiveMQProducer, 408
 - cms::MessageProducer, 2334
- getDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 959
 - activemq::core::ActiveMQProducer, 408
 - cms::MessageProducer, 2334
- getDouble
 - activemq::commands::ActiveMQMapMessage, 314
 - activemq::util::PrimitiveList, 2529
 - activemq::util::PrimitiveMap, 2540
 - activemq::util::PrimitiveValueNode, 2562
 - cms::MapMessage, 2109
 - decaf::internal::nio::ByteBuffer, 881
 - decaf::internal::util::ByteArrayAdapter, 857
 - decaf::nio::ByteBuffer, 924, 925
- getDoubleArray
 - decaf::internal::util::ByteArrayAdapter, 857
- getDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 858
- getDoubleCapacity
 - decaf::internal::util::ByteArrayAdapter, 858
- getDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 373
 - activemq::wireformat::openwire::utils::MessagePropertyInterco, 2341
 - cms::Message, 2174
- getEncoded
 - decaf::security::auth::x500::X500Principal, 3406
 - decaf::security::cert::Certificate, 969
 - decaf::security::Key, 1954
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX500Principal, 2440
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2444
- getEnumeration
 - cms::QueueBrowser, 2675
- getenv
 - decaf::lang::System, 3146
- getErrorCode

- decaf::net::SocketError, 2971
- getErrorString
 - decaf::net::SocketError, 2971
- getException
 - activemq::commands::ConnectionError, 1162
 - activemq::commands::ExceptionResponse, 1584
- getExceptionClass
 - activemq::commands::BrokerError, 747
- getExceptionListener
 - activemq::core::ActiveMQConnection, 239
 - activemq::core::ActiveMQSession, 454
 - cms::Connection, 1133
- getExpiration
 - activemq::commands::Message, 2153
- getFilter
 - decaf::util::logging::Handler, 1710
 - decaf::util::logging::Logger, 2032
 - decaf::util::logging::StreamHandler, 3078
- getFirstMessageId
 - activemq::commands::MessageAck, 2191
- getFirstNakNumber
 - activemq::commands::ReplayCommand, 2754
- getFloat
 - activemq::commands::ActiveMQMapMessage, 314
 - activemq::util::PrimitiveList, 2530
 - activemq::util::PrimitiveMap, 2540
 - activemq::util::PrimitiveValueNode, 2563
 - cms::MapMessage, 2110
 - decaf::internal::nio::ByteBuffer, 881, 882
 - decaf::internal::util::ByteArrayAdapter, 858
 - decaf::nio::ByteBuffer, 925
- getFloatArray
 - decaf::internal::util::ByteArrayAdapter, 859
- getFloatAt
 - decaf::internal::util::ByteArrayAdapter, 859
- getFloatCapacity
 - decaf::internal::util::ByteArrayAdapter, 859
- getFloatProperty
 - activemq::commands::ActiveMQMessageTemplate, 373
 - activemq::wireformat::openwire::utils::MessageProperty, 2341
 - cms::Message, 2175
- getFormat
 - decaf::security::Key, 1954
- getFormatId
 - activemq::commands::XATransactionId, 3412
- getFormatter
 - decaf::util::logging::Handler, 1710
 - decaf::util::logging::StreamHandler, 3078
- getFragment
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3341
 - decaf::net::URI, 3312
- getFreeThreadCount
 - decaf::util::concurrent::ThreadPool, 3177
- getGlobalPool
 - decaf::internal::AprPool, 628
 - decaf::internal::DecafRuntime, 1472
- getGlobalTransactionId
 - activemq::commands::XATransactionId, 3413
- getGroupID
 - activemq::commands::Message, 2153
- getGroupSequence
 - activemq::commands::Message, 2153
- getHead
 - decaf::util::logging::Formatter, 1700
 - decaf::util::logging::SimpleFormatter, 2961
- getHoldCount
 - decaf::util::concurrent::locks::ReentrantLock, 2693
- getHost
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3341
 - decaf::net::URI, 3312
- getId
 - activemq::state::TransactionState, 3266
- getIndex
 - decaf::net::URISyntaxException, 3337
- getInfo
 - activemq::state::ConnectionState, 1242
 - activemq::state::ConsumerState, 1329
 - activemq::state::ProducerState, 2656
 - activemq::state::SessionState, 2904
- getInitialDelayTime
 - activemq::transport::inactivity::InactivityMonitor, 1734
- getInitialReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1616
- getInput
 - decaf::net::URISyntaxException, 3338
- getInterrupt
 - decaf::io::Reader, 2683
 - decaf::net::BufferedSocket, 819
 - decaf::net::Socket, 2965
 - decaf::net::TcpSocket, 3155

- getInstance
 - activemq::transport::mock::MockTransport, 2374
 - activemq::transport::TransportRegistry, 3292
 - activemq::wireformat::WireFormatRegistry, 3401
 - decaf::security_-
 - provider::SecurityProviderMap, 2823
 - decaf::util::concurrent::ThreadPool, 3178
 - decaf::util::logging::LogManager, 2048
 - decaf::util::logging::LogWriter, 2055
- getInt
 - activemq::commands::ActiveMQMapMessage, 314
 - activemq::util::PrimitiveList, 2530
 - activemq::util::PrimitiveMap, 2541
 - activemq::util::PrimitiveValueNode, 2563
 - cms::MapMessage, 2110
 - decaf::internal::nio::ByteBuffer, 882
 - decaf::internal::util::ByteArrayAdapter, 859
 - decaf::nio::ByteBuffer, 926
- getIntArray
 - decaf::internal::util::ByteArrayAdapter, 860
- getIntAt
 - decaf::internal::util::ByteArrayAdapter, 860
- getIntCapacity
 - decaf::internal::util::ByteArrayAdapter, 860
- getIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 374
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2342
 - cms::Message, 2175
- getIssuerUniqueID
 - decaf::security::cert::X509Certificate, 3408
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2444
- getIssuerX500Principal
 - decaf::security::cert::X509Certificate, 3408
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2444
- getKeepAlive
 - decaf::net::BufferedSocket, 819
 - decaf::net::Socket, 2966
 - decaf::net::TcpSocket, 3155
- getKey
 - decaf::util::Map::Entry, 1570
- getKeyUsage
 - decaf::security::cert::X509Certificate, 3408
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2444
- getLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2705
 - activemq::core::ActiveMQConsumer, 268
 - activemq::core::ActiveMQSession, 455
- getLastMessageId
 - activemq::commands::MessageAck, 2191
- getLastNakNumber
 - activemq::commands::ReplayCommand, 2754
- getLastSequenceId
 - activemq::util::LongSequenceGenerator, 2090
- getLeastSignificantBits
 - decaf::util::UUID, 3359
- getLevel
 - decaf::util::logging::Handler, 1710
 - decaf::util::logging::Logger, 2032
 - decaf::util::logging::LogRecord, 2051
 - decaf::util::logging::StreamHandler, 3079
- getLimit
 - activemq::util::MemoryUsage, 2142
- getList
 - activemq::util::PrimitiveValueNode, 2563
- getLogger
 - decaf::util::logging::Logger, 2032
 - decaf::util::logging::LogManager, 2048
- getLoggerName
 - decaf::util::logging::LogRecord, 2051
- getLoggerNames
 - decaf::util::logging::LogManager, 2048
- getLong
 - activemq::commands::ActiveMQMapMessage, 315
 - activemq::util::PrimitiveList, 2530
 - activemq::util::PrimitiveMap, 2541
 - activemq::util::PrimitiveValueNode, 2563
 - cms::MapMessage, 2110
 - decaf::internal::nio::ByteBuffer, 883
 - decaf::internal::util::ByteArrayAdapter, 861
 - decaf::nio::ByteBuffer, 926, 927
- getLongArray
 - decaf::internal::util::ByteArrayAdapter, 861
- getLongAt
 - decaf::internal::util::ByteArrayAdapter, 861
- getLongCapacity

- decaf::internal::util::ByteArrayAdapter, 862
- getLongProperty
 - activemq::commands::ActiveMQMessageTemplate, 374
 - activemq::wireformat::openwire::utils::MessageProperties, 2342
 - cms::Message, 2176
- getMagic
 - activemq::commands::WireFormatInfo, 3373
- getMap
 - activemq::commands::ActiveMQMapMessage, 315
 - activemq::util::PrimitiveValueNode, 2564
- getMapNames
 - activemq::commands::ActiveMQMapMessage, 315
 - cms::MapMessage, 2111
- getMarshaledForm
 - activemq::commands::BaseDataStructure, 717
 - activemq::wireformat::MarshalAware, 2119
- getMarshaledProperties
 - activemq::commands::Message, 2153
 - activemq::commands::WireFormatInfo, 3373
- getMaxCacheSize
 - activemq::state::ConnectionStateTracker, 1246
 - activemq::transport::failover::FailoverTransport, 1616
- getMaximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1304
- getMaximumRedeliveries
 - activemq::core::ActiveMQTransactionContext, 624
- getMaxInactivityDuration
 - activemq::commands::WireFormatInfo, 3373
 - activemq::wireformat::openwire::OpenWireFormat, 2452
- getMaxInactivityDurationInitialDelay
 - activemq::commands::WireFormatInfo, 3373
- getMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2452
- getMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1616
- getMaxReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1616
- getMaxThreads
 - decaf::util::concurrent::ThreadPool, 3178
- getMessage
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2691
 - activemq::commands::BrokerError, 747
 - activemq::commands::JournalTrace, 1885
 - activemq::commands::MessageDispatch, 2222
 - activemq::core::DispatchData, 1535
 - cms::CMSException, 1032
 - decaf::lang::Exception, 1578
 - decaf::lang::Throwable, 3183
 - decaf::util::logging::LogRecord, 2051
- getMessageAck
 - activemq::commands::JournalQueueAck, 1836
- getMessageCount
 - activemq::commands::MessageAck, 2191
- getMessageId
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::Message, 2153
 - activemq::commands::MessageDispatchNotification, 2254
 - activemq::commands::MessagePull, 2349
- getMessageListener
 - activemq::cmsutil::CachedConsumer, 954
 - activemq::core::ActiveMQConsumer, 268
 - cms::MessageConsumer, 2215
- getMessageProperties
 - activemq::commands::Message, 2153
- getMessageSelector
 - activemq::cmsutil::CachedConsumer, 954
 - activemq::core::ActiveMQConsumer, 268
 - cms::MessageConsumer, 2215
 - cms::QueueBrowser, 2675
- getMessageSequenceId
 - activemq::commands::JournalTopicAck, 1861
- getMetadata
 - activemq::core::ActiveMQConnection, 239
 - cms::Connection, 1133
- getMimeType
 - activemq::commands::ActiveMQBlobMessage, 174
- getMostSignificantBits
 - decaf::util::UUID, 3359
- getName
 - activemq::commands::ActiveMQBlobMessage, 174
 - decaf::security::auth::x500::X500Principal, 3406
 - decaf::security::Principal, 2569

- decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2440
- decaf::util::logging::Logger, 2032
- getNetworkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2395
- getNetworkConsumerPath
 - activemq::commands::ConsumerInfo, 1304
- getNetworkProperties
 - activemq::commands::BrokerInfo, 778, 779
- getNetworkTTL
 - activemq::commands::NetworkBridgeFilter, 2395
- getNextLocalTransactionId
 - activemq::core::ActiveMQConnectionSupport, 255
- getNextSequenceId
 - activemq::util::LongSequenceGenerator, 2090
- getNextSessionId
 - activemq::core::ActiveMQConnectionSupport, 255
- getNextTempDestinationId
 - activemq::core::ActiveMQConnectionSupport, 255
- getNotAfter
 - decaf::security::cert::X509Certificate, 3408
- decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2444
- getNotBefore
 - decaf::security::cert::X509Certificate, 3408
- decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2444
- getNumReceivedMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2374
- getNumReceivedMessages
 - activemq::transport::mock::MockTransport, 2374
- getNumSentKeepAlives
 - activemq::transport::mock::MockTransport, 2374
- getNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::MockTransport, 2374
- getNumSentMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2374
- getNumSentMessages
 - activemq::transport::mock::MockTransport, 2374
- getObjectId
 - activemq::commands::RemoveInfo, 2705
- getOperationType
 - activemq::commands::DestinationInfo, 1487
- getOptions
 - activemq::commands::ActiveMQDestination, 279
- getOrderedTarget
 - activemq::commands::ActiveMQDestination, 279
- getOriginalDestination
 - activemq::commands::Message, 2154
- getOriginalTransactionId
 - activemq::commands::Message, 2154
- getOutputStream
 - decaf::io::Writer, 3404
 - decaf::net::BufferedSocket, 819
 - decaf::net::Socket, 2966
 - decaf::net::TcpSocket, 3155
 - decaf::util::logging::StreamHandler, 3079
- getParameters
 - activemq::util::CompositeData, 1089
- getParentId
 - activemq::commands::ConsumerId, 1277
 - activemq::commands::ProducerId, 2608
 - activemq::commands::SessionId, 2855
- getPassword
 - activemq::commands::ConnectionInfo, 1213, 1214
 - activemq::core::ActiveMQConnectionFactory, 247
 - activemq::core::ActiveMQConnectionSupport, 255
- getPeerBrokerInfos
 - activemq::commands::BrokerInfo, 779
- getPhysicalName
 - activemq::commands::ActiveMQDestination, 279
- getPooledThreadListener
 - decaf::util::concurrent::PooledThread, 2518
- getPoolSize
 - decaf::util::concurrent::ThreadPool, 3178
- getPort
 - decaf::internal::net::URIType, 3341
 - decaf::net::URI, 3313
- getPreferredWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2452
- getPrefetch

- activemq::commands::ConsumerControl, 1252
- getPrefetchSize
 - activemq::commands::ConsumerInfo, 1304
- getPreparedResult
 - activemq::state::TransactionState, 3266
- getPriority
 - activemq::cmsutil::CachedProducer, 959
 - activemq::cmsutil::CmsTemplate, 1046
 - activemq::commands::ConsumerInfo, 1304
 - activemq::commands::Message, 2154
 - activemq::core::ActiveMQProducer, 408
 - cms::MessageProducer, 2334
- getProducerId
 - activemq::commands::Message, 2154
 - activemq::commands::MessageId, 2281, 2282
 - activemq::commands::ProducerAck, 2581
 - activemq::commands::ProducerInfo, 2634
 - activemq::core::ActiveMQProducer, 409
- getProducerInfo
 - activemq::core::ActiveMQProducer, 409
- getProducerSequenceId
 - activemq::commands::MessageId, 2282
- getProducerState
 - activemq::state::SessionState, 2904
- getProducerStates
 - activemq::state::SessionState, 2904
- getProducerWindowSize
 - activemq::core::ActiveMQConnectionSupport, 256
- getProperties
 - activemq::commands::WireFormatInfo, 3374
 - activemq::core::ActiveMQConnectionSupport, 256
 - activemq::util::ActiveMQProperties, 415, 416
 - activemq::wireformat::stomp::StompFrame, 3063
 - decaf::util::logging::LogManager, 2048
- getProperty
 - activemq::util::ActiveMQProperties, 416
 - activemq::wireformat::stomp::StompFrame, 3064
 - cms::CMSProperties, 1037
 - decaf::util::logging::LogManager, 2048
 - decaf::util::Properties, 2659, 2660
- getPropertyNames
 - activemq::commands::ActiveMQMessageTemplate, 374
 - cms::Message, 2176
- getProvider
 - decaf::security_ -
 - provider::SecurityProviderRegistrar, 2826
 - getProviderMajorVersion
 - activemq::core::ActiveMQConnectionMetaData, 251
 - cms::ConnectionMetaData, 1239
 - getProviderMinorVersion
 - activemq::core::ActiveMQConnectionMetaData, 251
 - cms::ConnectionMetaData, 1239
 - getProviderVersion
 - activemq::core::ActiveMQConnectionMetaData, 252
 - cms::ConnectionMetaData, 1240
 - getPublicKey
 - decaf::security::cert::Certificate, 969
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2444, 2445
 - getQuery
 - decaf::internal::net::URIType, 3342
 - decaf::net::URI, 3313
 - getQueue
 - cms::QueueBrowser, 2676
 - getQueueName
 - activemq::commands::ActiveMQQueue, 422
 - activemq::commands::ActiveMQTempQueue, 526
 - cms::Queue, 2674
 - cms::TemporaryQueue, 3166
 - getRawAuthority
 - decaf::net::URI, 3313
 - getRawFragment
 - decaf::net::URI, 3313
 - getRawPath
 - decaf::net::URI, 3313
 - getRawQuery
 - decaf::net::URI, 3314
 - getRawSchemeSpecificPart
 - decaf::net::URI, 3314
 - getRawUserInfo
 - decaf::net::URI, 3314
 - getReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1734
 - getReason
 - decaf::net::URISyntaxException, 3338
 - getReceiveBufferSize
 - decaf::net::BufferedSocket, 819
 - decaf::net::Socket, 2966
 - decaf::net::TcpSocket, 3155
 - getReceiveTimeout

- activemq::cmsutil::CmsTemplate, 1046
- getReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1616
- getRedeliveryCounter
 - activemq::commands::Message, 2154
 - activemq::commands::MessageDispatch, 2222
- getRedeliveryDelay
 - activemq::core::ActiveMQTransactionContext, 624
- getReferences
 - decaf::internal::nio::ByteArrayPerspective, 911
- getRemoteAddress
 - activemq::transport::failover::FailoverTransport, 1616
 - activemq::transport::IOTransport, 1825
 - activemq::transport::mock::MockTransport, 2374
 - activemq::transport::Transport, 3274
 - activemq::transport::TransportFilter, 3283
- getRemoteBlobUrl
 - activemq::commands::ActiveMQBlobMessage, 174
- getReplyTo
 - activemq::commands::Message, 2154
- getResourceLifecycleManager
 - activemq::cmsutil::CmsAccessor, 1026
 - activemq::cmsutil::SessionPool, 2902
- getResponse
 - activemq::transport::correlator::FutureResponse, 1704, 1705
- getResult
 - activemq::commands::IntegerResponse, 1779
- getReuseAddress
 - decaf::net::BufferedSocket, 820
 - decaf::net::Socket, 2966
 - decaf::net::TcpSocket, 3156
- getRuntime
 - decaf::lang::Runtime, 2817
- getSafeValue
 - decaf::util::StlQueue, 3046
- getScheme
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3342
 - decaf::net::URI, 3314
- getSchemeSpecificPart
 - decaf::internal::net::URIType, 3342
 - decaf::net::URI, 3314
- getSecurityProviderNames
 - decaf::security_-
provider::SecurityProviderMap, 2823
- getSelector
 - activemq::commands::ConsumerInfo, 1304
 - activemq::commands::SubscriptionInfo, 3100
- getSendBufferSize
 - decaf::net::BufferedSocket, 820
 - decaf::net::Socket, 2967
 - decaf::net::TcpSocket, 3156
- getSendTimeout
 - activemq::core::ActiveMQConnectionSupport, 256
 - activemq::core::ActiveMQProducer, 409
- getServiceName
 - activemq::commands::DiscoveryEvent, 1513
- getSession
 - activemq::cmsutil::PooledSession, 2515
- getSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 1026
- getSessionId
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionInfo, 2879
 - activemq::core::ActiveMQSession, 455
- getSessionInfo
 - activemq::core::ActiveMQSession, 455
- getSessionState
 - activemq::state::ConnectionState, 1242
- getSessionStates
 - activemq::state::ConnectionState, 1242
- getShort
 - activemq::commands::ActiveMQMapMessage, 315
 - activemq::util::PrimitiveList, 2531
 - activemq::util::PrimitiveMap, 2542
 - activemq::util::PrimitiveValueNode, 2564
 - cms::MapMessage, 2111
 - decaf::internal::nio::ByteBuffer, 883, 884
 - decaf::internal::util::ByteArrayAdapter, 862
 - decaf::nio::ByteBuffer, 927
- getShortArray
 - decaf::internal::util::ByteArrayAdapter, 862
- getShortAt
 - decaf::internal::util::ByteArrayAdapter, 862
- getShortCapacity
 - decaf::internal::util::ByteArrayAdapter, 863
- getShortProperty
 - activemq::commands::ActiveMQMessageTemplate, 375

- activemq::wireformat::openwire::utils::MessagePropertyException, 1578
- 2342
- cms::Message, 2177
- getSigAlgName
 - decaf::security::cert::X509Certificate, 3408
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2445
- getSigAlgOID
 - decaf::security::cert::X509Certificate, 3408
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2445
- getSigAlgParams
 - decaf::security::cert::X509Certificate, 3409
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2445
- getSignature
 - decaf::security::cert::X509Certificate, 3409
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2445
- getSize
 - activemq::commands::ActiveMQTextMessage, 575
 - activemq::commands::Message, 2154
 - activemq::commands::ProducerAck, 2581
- getSocketHandle
 - decaf::net::TcpSocket, 3156
- getSoLinger
 - decaf::net::BufferedSocket, 820
 - decaf::net::Socket, 2967
 - decaf::net::TcpSocket, 3156
- getSoTimeout
 - decaf::net::BufferedSocket, 821
 - decaf::net::Socket, 2967
 - decaf::net::TcpSocket, 3156
- getSource
 - decaf::internal::net::URIType, 3342
- getSourceFile
 - decaf::util::logging::LogRecord, 2051
- getSourceFunction
 - decaf::util::logging::LogRecord, 2051
- getSourceLine
 - decaf::util::logging::LogRecord, 2052
- getStackTrace
 - cms::CMSException, 1032
 - decaf::lang::Exception, 1578
 - decaf::lang::Throwable, 3183
- getStackTraceElements
 - activemq::commands::BrokerError, 747
- getStackTraceString
 - cms::CMSException, 1033
- decaf::lang::Exception, 1578
- decaf::lang::Throwable, 3183
- getString
 - activemq::commands::ActiveMQMapMessage, 316
 - activemq::util::PrimitiveList, 2531
 - activemq::util::PrimitiveMap, 2542
 - activemq::util::PrimitiveValueNode, 2564
 - cms::MapMessage, 2111
- getStringProperty
 - activemq::commands::ActiveMQMessageTemplate, 2342
 - activemq::wireformat::openwire::utils::MessagePropertyInterface, 2342
 - cms::Message, 2177
- getSubscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 2729, 2730
 - activemq::commands::SubscriptionInfo, 3100
- getSubjectUniqueID
 - decaf::security::cert::X509Certificate, 3409
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2445
- getSubjectX500Principal
 - decaf::security::cert::X509Certificate, 3409
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2446
- getSubscribedDestination
 - activemq::commands::SubscriptionInfo, 3100
- getSubscriptionName
 - activemq::commands::ConsumerInfo, 1304
- getSubscriptionName
 - activemq::commands::JournalTopicAck, 1861
- getTail
 - decaf::util::logging::Formatter, 1700
 - decaf::util::logging::SimpleFormatter, 2961
- getTargetConsumerId
 - activemq::commands::Message, 2154, 2155
- getTBSCertificate
 - decaf::security::cert::X509Certificate, 3409
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2446
- getTcpNoDelay
 - decaf::net::TcpSocket, 3157
- getTempDesinations
 - activemq::state::ConnectionState, 1242
- getText

- activemq::commands::ActiveMQTextMessage, 576
- cms::TextMessage, 3170
- getTime
 - decaf::util::Date, 1469
- getTimeout
 - activemq::commands::DestinationInfo, 1487
 - activemq::commands::MessagePull, 2349
 - activemq::transport::failover::FailoverTransport, 1616
- getTimeStamp
 - activemq::commands::Message, 2155
 - decaf::util::logging::LogRecord, 2052
- getTimeToLive
 - activemq::cmsutil::CachedProducer, 959
 - activemq::cmsutil::CmsTemplate, 1046
 - activemq::core::ActiveMQProducer, 409
 - cms::MessageProducer, 2334
- getTopicName
 - activemq::commands::ActiveMQTempTopic, 551
 - activemq::commands::ActiveMQTopic, 601
 - cms::TemporaryTopic, 3168
 - cms::Topic, 3215
- getTransactionId
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::JournalTransaction, 1908
 - activemq::commands::Message, 2155
 - activemq::commands::MessageAck, 2191
 - activemq::commands::TransactionInfo, 3242
 - activemq::core::ActiveMQTransactionContext, 624
- getTransactionState
 - activemq::state::ConnectionState, 1242
- getTransactionStates
 - activemq::state::ConnectionState, 1242
- getTransport
 - activemq::core::ActiveMQConnectionSupportgetValue, 256
 - activemq::transport::failover::BackupTransport, 644
- getTransportListener
 - activemq::transport::failover::FailoverTransport, 1616
 - activemq::transport::IOTransport, 1825
 - activemq::transport::mock::MockTransport, 2374
 - activemq::transport::Transport, 3274
 - activemq::transport::TransportFilter, 3284
- getTransportNames
 - activemq::transport::TransportRegistry, 3292
- getTreadId
 - decaf::util::logging::LogRecord, 2052
- getType
 - activemq::commands::JournalTransaction, 1908
 - activemq::commands::Message, 2155
 - activemq::commands::TransactionInfo, 3242
 - activemq::util::PrimitiveValueNode, 2564
 - decaf::security::cert::Certificate, 970
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2446
- getUnconsumedMessages
 - activemq::core::ActiveMQSessionExecutor, 462
- getURI
 - activemq::transport::failover::URIPool, 3330
- getUri
 - activemq::transport::failover::BackupTransport, 645
- getUsage
 - activemq::util::MemoryUsage, 2142
- getUseParentHandlers
 - decaf::util::logging::Logger, 2032
- getUserID
 - activemq::commands::Message, 2155
- getUserInfo
 - decaf::internal::net::URIType, 3342
 - decaf::net::URI, 3314
- getUserName
 - activemq::commands::ConnectionInfo, 1214
- getUsername
 - activemq::core::ActiveMQConnectionFactory, 247
 - activemq::core::ActiveMQConnectionSupport, 256
- getValue
 - activemq::commands::BrokerId, 753
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::LocalTransactionId, 1995
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionId, 2855
 - activemq::util::PrimitiveValueNode, 2565
 - decaf::util::Map::Entry, 1570
- getVersion
 - activemq::commands::WireFormatInfo, 3374

- activemq::wireformat::openwire::OpenWireFormat, 2453
- activemq::wireformat::stomp::StompWireFormat, 3072
- activemq::wireformat::WireFormat, 3364
- decaf::security::cert::X509Certificate, 3409
- decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2446
- getWasPrepared
 - activemq::commands::JournalTransaction, 1908
- getWhen
 - decaf::util::TimerTask, 3200
- getWindowSize
 - activemq::commands::ProducerInfo, 2634
- getWireFormat
 - activemq::transport::mock::MockTransport, 2375
- getWireFormatNames
 - activemq::wireformat::WireFormatRegistry, 3401
- getWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1734
- getX509Name
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2441
- globalTransactionId
 - activemq::commands::XATransactionId, 3414
- groupID
 - activemq::commands::Message, 2161
- groupSequence
 - activemq::commands::Message, 2161
- handleTransportFailure
 - activemq::transport::failover::FailoverTransport, 1616
- hasArray
 - decaf::internal::nio::ByteBuffer, 884
 - decaf::internal::nio::CharArrayBuffer, 995
 - decaf::internal::nio::DoubleArrayBuffer, 1553
 - decaf::internal::nio::FloatArrayBuffer, 1663
 - decaf::internal::nio::IntArrayBuffer, 1749
 - decaf::internal::nio::LongArrayBuffer, 2076
 - decaf::internal::nio::ShortArrayBuffer, 2920
 - decaf::nio::ByteBuffer, 928
 - decaf::nio::CharBuffer, 1007
 - decaf::nio::DoubleBuffer, 1562
 - decaf::nio::FloatBuffer, 1671
- decaf::nio::IntBuffer, 1757
- decaf::nio::LongBuffer, 2085
- decaf::nio::ShortBuffer, 2929
- hashCode
- decaf::security::auth::x500::X500Principal, 3406
- hasMoreTokens
- decaf::util::StringTokenizer, 3095
- hasNegotiator
 - activemq::wireformat::openwire::OpenWireFormat, 2453
 - activemq::wireformat::stomp::StompWireFormat, 3072
 - activemq::wireformat::WireFormat, 3364
- hasNext
 - decaf::util::Iterator, 1832
- hasPrevious
 - decaf::util::ListIterator, 1991
- hasProperty
 - activemq::util::ActiveMQProperties, 416
 - activemq::wireformat::stomp::StompFrame, 3064
 - cms::CMSProperties, 1038
 - decaf::util::Properties, 2660
- hasRemaining
 - decaf::nio::Buffer, 806
- hasUnconsumedMessages
- activemq::core::ActiveMQSessionExecutor, 462
- HAVE_PTHREAD_H
 - activemq/util/Config.h, 3589
- HAVE_UUID_T
 - activemq/util/Config.h, 3589
- HAVE_UUID_UUID_H
 - activemq/util/Config.h, 3589
- HEADER_ACK
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- HEADER_CLIENT_ID
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- HEADER_CONSUMERPRIORITY
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- HEADER_CONTENTLENGTH
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- HEADER_CORRELATIONID
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- HEADER_DESTINATION
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- HEADER_DISPATCH_ASYNC

activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_EXCLUSIVE	HEADER_REQUESTID
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_EXPIRES	HEADER_RESPONSEID
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_ID	HEADER_RETROACTIVE
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_JMSPRIORITY	HEADER_SELECTOR
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_LOGIN	HEADER_SESSIONID
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_MAXPENDINGMSGLIMIT	HEADER_SUBSCRIPTION
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_MESSAGE	HEADER_SUBSCRIPTIONNAME
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_MESSAGEID	HEADER_TIMESTAMP
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_NOLOCAL	HEADER_TRANSACTIONID
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_OLDSUBSCRIPTIONNAME	HEADER_TRANSFORMATION
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_PASSWORD	HEADER_TRANSFORMATION_ERROR
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_PERSISTENT	HEADER_TYPE
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::stomp::StompCommandConstants,
3059	3059
HEADER_PREFETCHSIZE	HexStringParser
activemq::wireformat::stomp::StompCommandConstants,	decaf::internal::util::HexStringParser, 1713
3059	
HEADER_RECEIPT_REQUIRED	HexTable
activemq::wireformat::stomp::StompCommandConstants,	activemq::wireformat::openwire::utils::HexTable,
3059	1715
HEADER_RECEIPTID	highestOneBit
activemq::wireformat::stomp::StompCommandConstants,	decaf::lang::Integer, 1767
3059	decaf::lang::Long, 2062
HEADER_REDELIVERED	HOURS
activemq::wireformat::stomp::StompCommandConstants,	decaf::util::concurrent::TimeUnit, 3213
3059	HttpRetryException
HEADER_REDELIVERYCOUNT	decaf::net::HttpRetryException, 1717,
activemq::wireformat::stomp::StompCommandConstants,	1718
3059	
HEADER_REPLYTO	ID_ACTIVEMQBLOBMESSAGE

activemq::commands::ActiveMQBlobMessage, 176	activemq::commands::ConsumerControl, 1254
ID_ ACTIVEMQBYTESMESSAGE	ID_ CONSUMERID
activemq::commands::ActiveMQBytesMessage, 212	activemq::commands::ConsumerId, 1278
ID_ ACTIVEMQDESTINATION	ID_ CONSUMERINFO
activemq::commands::ActiveMQDestination, ID_	activemq::commands::ConsumerInfo, 1307
284	CONTROLCOMMAND
ID_ ACTIVEMQMAPMESSAGE	activemq::commands::ControlCommand, 1333
activemq::commands::ActiveMQMapMessage, ID_	DATAARRAYRESPONSE
320	activemq::commands::DataArrayResponse, 1358
ID_ ACTIVEMQMESSAGE	ID_ DATARESPONSE
activemq::commands::ActiveMQMessage, 344	activemq::commands::DataResponse, 1397
ID_ ACTIVEMQOBJECTMESSAGE	ID_ DESTINATIONINFO
activemq::commands::ActiveMQObjectMessage, 385	activemq::commands::DestinationInfo, 1488
ID_ ACTIVEMQQUEUE	ID_ DISCOVERYEVENT
activemq::commands::ActiveMQQueue, 422	activemq::commands::DiscoveryEvent, 1514
ID_ ACTIVEMQSTREAMMESSAGE	ID_ EXCEPTIONRESPONSE
activemq::commands::ActiveMQStreamMessage, 478	activemq::commands::ExceptionResponse, 1584
ID_ ACTIVEMQTEMPDESTINATION	ID_ FLUSHCOMMAND
activemq::commands::ActiveMQTempDestination, ID_	activemq::commands::FlushCommand, 1678
502	ID_ INTEGERRESPONSE
ID_ ACTIVEMQTEMPQUEUE	activemq::commands::IntegerResponse, 1779
activemq::commands::ActiveMQTempQueue, 527	ID_ JOURNALQUEUEACK
ID_ ACTIVEMQTEMPTOPIC	activemq::commands::JournalQueueAck, 1837
activemq::commands::ActiveMQTempTopic, 552	ID_ JOURNALTOPICACK
ID_ ACTIVEMQTEXTMESSAGE	activemq::commands::JournalTopicAck, 1862
activemq::commands::ActiveMQTextMessage, 577	ID_ JOURNALTRACE
ID_ ACTIVEMQTOPIC	activemq::commands::JournalTrace, 1885
activemq::commands::ActiveMQTopic, 601	ID_ JOURNALTRANSACTION
ID_ BROKERID	activemq::commands::JournalTransaction, 1909
activemq::commands::BrokerId, 753	ID_ KEEPALIVEINFO
ID_ BROKERINFO	activemq::commands::KeepAliveInfo, 1932
activemq::commands::BrokerInfo, 782	ID_ LASTPARTIALCOMMAND
ID_ CONNECTIONCONTROL	activemq::commands::LastPartialCommand, 1960
activemq::commands::ConnectionControl, 1139	ID_ LOCALTRANSACTIONID
ID_ CONNECTIONERROR	activemq::commands::LocalTransactionId, 1996
activemq::commands::ConnectionError, 1163	ID_ MESSAGE
ID_ CONNECTIONID	activemq::commands::Message, 2161
activemq::commands::ConnectionId, 1190	ID_ MESSAGEACK
ID_ CONNECTIONINFO	activemq::commands::MessageAck, 2193
activemq::commands::ConnectionInfo, 1216	ID_ MESSAGEDISPATCH
ID_ CONSUMERCONTROL	

- activemq::commands::MessageDispatch, 2223
- ID_ MESSAGEDISPATCHNOTIFICATION
- activemq::commands::MessageDispatchNotification, 2256
- ID_ MESSAGEID
- activemq::commands::MessageId, 2283
- ID_ MESSAGEPULL
- activemq::commands::MessagePull, 2350
- ID_ NETWORKBRIDGEFILTER
- activemq::commands::NetworkBridgeFilter, 2396
- ID_ PARTIALCOMMAND
- activemq::commands::PartialCommand, 2476
- ID_ PRODUCERACK
- activemq::commands::ProducerAck, 2582
- ID_ PRODUCERID
- activemq::commands::ProducerId, 2609
- ID_ PRODUCERINFO
- activemq::commands::ProducerInfo, 2635
- ID_ REMOVEINFO
- activemq::commands::RemoveInfo, 2706
- ID_ REMOVESUBSCRIPTIONINFO
- activemq::commands::RemoveSubscriptionInfo, 2731
- ID_ REPLAYCOMMAND
- activemq::commands::ReplayCommand, 2755
- ID_ RESPONSE
- activemq::commands::Response, 2784
- ID_ SESSIONID
- activemq::commands::SessionId, 2856
- ID_ SESSIONINFO
- activemq::commands::SessionInfo, 2880
- ID_ SHUTDOWNINFO
- activemq::commands::ShutdownInfo, 2936
- ID_ SUBSCRIPTIONINFO
- activemq::commands::SubscriptionInfo, 3101
- ID_ TRANSACTIONID
- activemq::commands::TransactionId, 3219
- ID_ TRANSACTIONINFO
- activemq::commands::TransactionInfo, 3243
- ID_ WIREFORMATINFO
- activemq::commands::WireFormatInfo, 3378
- ID_ XATRANSACTIONID
- activemq::commands::XATransactionId, 3414
- IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1720, 1721
- IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1723, 1724
- IllegalStateException
 - cms::IllegalStateException, 1729
 - decaf::lang::exceptions::IllegalStateException, 1726, 1727
- IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1730, 1731
- InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1734
- increaseUsage
 - activemq::util::MemoryUsage, 2143
 - activemq::util::Usage, 3352
- incrementAndGet
 - decaf::util::concurrent::atomic::AtomicInteger, 636
- indexOf
 - decaf::util::List, 1986
 - decaf::util::StlList, 3025
- IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1737, 1738
- INDIVIDUAL_ACKNOWLEDGE
 - cms::Session, 2842
- Info
 - decaf::util::logging, 144
- info
 - decaf::util::logging::Logger, 2033
 - decaf::util::logging::SimpleLogger, 2963
- init
 - activemq::cmsutil::CmsAccessor, 1027
 - activemq::cmsutil::CmsDestinationAccessor, 1029
 - activemq::cmsutil::CmsTemplate, 1046
 - activemq::cmsutil::DestinationResolver, 1509
 - activemq::cmsutil::DynamicDestinationResolver, 1568
- initCause
 - decaf::lang::Exception, 1578
 - decaf::lang::Throwable, 3183
- initializeLibrary
 - activemq::library::ActiveMQCPP, 272, 273
- initializeRuntime
 - decaf::lang::Runtime, 2817
- inputStream
 - decaf::io::FilterInputStream, 1639
- inReceive
 - activemq::wireformat::openwire::OpenWireFormat, 2453

activemq::wireformat::stomp::StompWireFormat, 3072	InvalidDestinationException, 1809
activemq::wireformat::WireFormat, 3365	InvalidKeyException
insert	decaf::security::InvalidKeyException, 1810, 1811
decaf::internal::util::TimerTaskHeap, 3203	InvalidMarkException
IntArrayBuffer	decaf::nio::InvalidMarkException, 1813, 1814
decaf::internal::nio::IntArrayBuffer, 1745, 1746	InvalidSelectorException
intBitsToFloat	cms::InvalidSelectorException, 1816
decaf::lang::Float, 1652	InvalidStateException
IntBuffer	decaf::lang::exceptions::InvalidStateException, 1817, 1818
decaf::nio::IntBuffer, 1753	IOException
Integer	decaf::io::IOException, 1820, 1821
decaf::lang::Integer, 1764	IOTransport
INTEGER_TYPE	activemq::transport::IOTransport, 1824
activemq::util::PrimitiveValueNode, 2559	isAbsolute
IntegerResponse	decaf::internal::net::URIType, 3342
activemq::commands::IntegerResponse, 1778	decaf::net::URI, 3315
IntegerResponseMarshaller	IntegerResponseMarshaller, 1785
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1785	activemq::commands::ActiveMQDestination, 280
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1781	isAlwaysSyncSend
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1789	getResponseViaActiveMQConnectionSupport, 257
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1793	IntegerResponseMarshaller, 1793
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1797	activemq::core::ActiveMQSession, 455
InternalCommandListener	isBroker
activemq::transport::mock::InternalCommandListener, 1800	activemq::transport::failover::FailoverTransport, 1617
InterruptedException	isBrokerInfo
decaf::lang::exceptions::InterruptedException, 1802, 1803	activemq::commands::BaseCommand, 652
InterruptedIOException	activemq::commands::BrokerInfo, 779
decaf::io::InterruptedIOException, 1805, 1806	activemq::commands::Command, 1064
int Value	isBrokerMasterConnector
activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2552	activemq::commands::ConnectionInfo, 1214
decaf::lang::Byte, 844	isBrowser
decaf::lang::Character, 986	activemq::commands::ConsumerInfo, 1304
decaf::lang::Double, 1542	isBusy
decaf::lang::Float, 1652	decaf::util::concurrent::PooledThread, 2519
decaf::lang::Integer, 1767	isCacheEnabled
decaf::lang::Long, 2062	activemq::commands::WireFormatInfo, 3374
decaf::lang::Number, 2433	activemq::wireformat::openwire::OpenWireFormat, 2453
decaf::lang::Short, 2910	isCancelled
decaf::util::concurrent::atomic::AtomicInteger, 637	decaf::util::concurrent::Future, 1703
InvalidClientIdException	isClientAcknowledge
cms::InvalidClientIdException, 1808	activemq::core::ActiveMQSession, 455
	isClientMaster

- activemq::commands::ConnectionInfo, 1214
- isClose
 - activemq::commands::ConnectionControl, 1137
 - activemq::commands::ConsumerControl, 1253
- isClosed
 - activemq::core::ActiveMQConnection, 239
 - activemq::core::ActiveMQConsumer, 268
 - activemq::core::ActiveMQProducer, 409
 - activemq::core::MessageDispatchChannel, 2226
 - activemq::transport::failover::BackupTransport, 645
 - activemq::transport::failover::FailoverTransport, 1617
 - activemq::transport::IOTransport, 1825
 - activemq::transport::mock::MockTransport, 2375
 - activemq::transport::tcp::TcpTransport, 3161
 - activemq::transport::Transport, 3274
 - activemq::transport::TransportFilter, 3284
 - decaf::io::FilterInputStream, 1633
 - decaf::io::FilterOutputStream, 1642
- isComposite
 - activemq::commands::ActiveMQDestination, 280
- isCompressed
 - activemq::commands::Message, 2155
- isConnected
 - activemq::transport::failover::FailoverTransport, 1617
 - activemq::transport::IOTransport, 1825
 - activemq::transport::mock::MockTransport, 2375
 - activemq::transport::tcp::TcpTransport, 3161
 - activemq::transport::Transport, 3275
 - activemq::transport::TransportFilter, 3284
 - decaf::net::BufferedSocket, 821
 - decaf::net::Socket, 2968
 - decaf::net::TcpSocket, 3157
- isConnectionAdvisory
 - activemq::commands::ActiveMQDestination, 280
- isConnectionInfo
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1064
 - activemq::commands::ConnectionInfo, 1214
- isConsumerAdvisory
 - activemq::commands::ActiveMQDestination, 280
- isConsumerInfo
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1064
 - activemq::commands::ConsumerInfo, 1304
- isDeletedByBroker
 - activemq::commands::ActiveMQBlobMessage, 175
- isDigit
 - decaf::lang::Character, 986
- isDispatchAsync
 - activemq::commands::ConsumerInfo, 1304
 - activemq::commands::ProducerInfo, 2634
- isDone
 - decaf::util::concurrent::Future, 1703
- isDroppable
 - activemq::commands::Message, 2155
- isDuplexConnection
 - activemq::commands::BrokerInfo, 779
- isDupsOkAcknowledge
 - activemq::core::ActiveMQSession, 455
- isEmpty
 - activemq::core::ActiveMQSessionExecutor, 462
 - activemq::core::MessageDispatchChannel, 2226
 - activemq::util::ActiveMQProperties, 416
 - cms::CMSProperties, 1038
 - decaf::internal::util::TimerTaskHeap, 3203
 - decaf::util::AbstractCollection, 153
 - decaf::util::Collection, 1059
 - decaf::util::concurrent::ConcurrentStlMap, 1109
 - decaf::util::concurrent::SynchronousQueue, 3141
 - decaf::util::Map, 2100
 - decaf::util::Properties, 2660
 - decaf::util::StlList, 3025
 - decaf::util::StlMap, 3037
 - decaf::util::StlSet, 3055
- isEnabled
 - activemq::transport::failover::BackupTransportPool, 648
- isExclusive
 - activemq::commands::ActiveMQDestination, 280
 - activemq::commands::ConsumerInfo, 1305
- isExit
 - activemq::commands::ConnectionControl, 1138
- isExpired
 - activemq::commands::Message, 2155
- isExplicitQosEnabled

- activemq::cmsutil::CmsTemplate, 1047
- isFailOnClose
 - activemq::transport::mock::MockTransport, 2375
- isFailOnKeepAliveSends
 - activemq::transport::mock::MockTransport, 2376
- isFailOnReceiveMessage
 - activemq::transport::mock::MockTransport, 2376
- isFailOnSendMessage
 - activemq::transport::mock::MockTransport, 2376
- isFailOnStart
 - activemq::transport::mock::MockTransport, 2376
- isFailOnStop
 - activemq::transport::mock::MockTransport, 2376
- isFair
 - decaf::util::concurrent::locks::ReentrantLock, 2694
 - decaf::util::concurrent::Semaphore, 2832
- isFaultTolerant
 - activemq::commands::ConnectionControl, 1138
 - activemq::transport::failover::FailoverTransport, 1617
 - activemq::transport::IOTransport, 1826
 - activemq::transport::mock::MockTransport, 2376
 - activemq::transport::tcp::TcpTransport, 3161
 - activemq::transport::Transport, 3275
 - activemq::transport::TransportFilter, 3284
- isFaultTolerantConfiguration
 - activemq::commands::BrokerInfo, 780
- isFlush
 - activemq::commands::ConsumerControl, 1253
- isFull
 - activemq::util::MemoryUsage, 2143
 - activemq::util::Usage, 3352
- isHeldByCurrentThread
 - decaf::util::concurrent::locks::ReentrantLock, 2694
- isIndividualAcknowledge
 - activemq::core::ActiveMQSession, 456
- isInfinite
 - decaf::lang::Double, 1542
 - decaf::lang::Float, 1653
- isInitialized
 - activemq::transport::failover::FailoverTransport, 1617
- isInTransaction
 - activemq::core::ActiveMQTransactionContext, 624
- isISOControl
 - decaf::lang::Character, 986
- isKeepAliveInfo
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1064
 - activemq::commands::KeepAliveInfo, 1932
- isKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1734
- isLetter
 - decaf::lang::Character, 986
- isLetterOrDigit
 - decaf::lang::Character, 986
- isLocked
 - decaf::util::concurrent::Lock, 2024
 - decaf::util::concurrent::locks::ReentrantLock, 2694
- isLoggable
 - decaf::util::logging::Filter, 1630
 - decaf::util::logging::Handler, 1710
 - decaf::util::logging::Logger, 2033
 - decaf::util::logging::StreamHandler, 3079
- isLowerCase
 - decaf::lang::Character, 986
- isManageable
 - activemq::commands::ConnectionInfo, 1214
- isMarshalAware
 - activemq::commands::ActiveMQMapMessage, 316
 - activemq::commands::BaseDataStructure, 717
 - activemq::commands::Message, 2155
 - activemq::commands::WireFormatInfo, 3374
 - activemq::wireformat::MarshalAware, 2119
- isMasterBroker
 - activemq::commands::BrokerInfo, 780
- isMessage
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1064
 - activemq::commands::Message, 2156
- isMessageAck
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1064
 - activemq::commands::MessageAck, 2191
- isMessageDispatch
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1065
 - activemq::commands::MessageDispatch, 2222

- isMessageDispatchNotification
 - activemq::commands::BaseCommand, 653
 - activemq::commands::Command, 1065
 - activemq::commands::MessageDispatchNotification, 2254
- isMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 1047
- isMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 1047
- isNaN
 - decaf::lang::Double, 1543
 - decaf::lang::Float, 1653
- isNetworkConnection
 - activemq::commands::BrokerInfo, 780
- isNetworkSubscription
 - activemq::commands::ConsumerInfo, 1305
- isNoLocal
 - activemq::cmsutil::CmsTemplate, 1047
 - activemq::commands::ConsumerInfo, 1305
- isNoRangeAcks
 - activemq::commands::ConsumerInfo, 1305
- isOpaque
 - decaf::internal::net::URIType, 3343
 - decaf::net::URI, 3315
- isOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1305
- isOrdered
 - activemq::commands::ActiveMQDestination, 280
- isPending
 - activemq::threads::CompositeTask, 1090
 - activemq::transport::failover::BackupTransportPool, 648
 - activemq::transport::failover::CloseTransportsTask, 1022
 - activemq::transport::failover::FailoverTransport, 1618
- isPersistent
 - activemq::commands::Message, 2156
- isPrepared
 - activemq::state::TransactionState, 3266
- isProducerAck
 - activemq::commands::BaseCommand, 654
 - activemq::commands::Command, 1065
 - activemq::commands::ProducerAck, 2581
- isProducerAdvisory
 - activemq::commands::ActiveMQDestination, 280
- isProducerInfo
 - activemq::commands::BaseCommand, 654
 - activemq::commands::Command, 1065
 - activemq::commands::ProducerInfo, 2634
- isPubSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 1029
 - isQueue
 - activemq::commands::ActiveMQDestination, 281
 - isRandomize
 - activemq::transport::failover::FailoverTransport, 1618
 - activemq::transport::failover::URIPool, 3330
 - isReadOnly
 - decaf::internal::nio::ByteBuffer, 884
 - decaf::internal::nio::CharArrayBuffer, 995
 - decaf::internal::nio::DoubleArrayBuffer, 1553
 - decaf::internal::nio::FloatArrayBuffer, 1663
 - decaf::internal::nio::IntArrayBuffer, 1749
 - decaf::internal::nio::LongArrayBuffer, 2076
 - decaf::internal::nio::ShortArrayBuffer, 2920
 - decaf::nio::Buffer, 806
 - decaf::nio::ByteBuffer, 928
 - isReadOnlyBody
 - activemq::commands::Message, 2156
 - isReadOnlyProperties
 - activemq::commands::Message, 2156
 - isRecievedByDFBridge
 - activemq::commands::Message, 2156
 - isRemoveInfo
 - activemq::commands::BaseCommand, 654
 - activemq::commands::Command, 1065
 - activemq::commands::RemoveInfo, 2705
 - isRemoveSubscriptionInfo
 - activemq::commands::BaseCommand, 654
 - activemq::commands::Command, 1065
 - activemq::commands::RemoveSubscriptionInfo, 2730
 - isResponse
 - activemq::commands::BaseCommand, 654
 - activemq::commands::Command, 1065
 - activemq::commands::Response, 2783
 - isResponseRequired
 - activemq::commands::BaseCommand, 654
 - activemq::commands::Command, 1065
 - isRestoreConsumers
 - activemq::state::ConnectionStateTracker, 1246
 - isRestoreProducers
 - activemq::state::ConnectionStateTracker, 1246
 - isRestoreSessions
 - activemq::state::ConnectionStateTracker, 1246
 - isRestoreTransaction

- activemq::state::ConnectionStateTracker, 1246
- isResume
 - activemq::commands::ConnectionControl, 1138
- isRetroactive
 - activemq::commands::ConsumerInfo, 1305
- isRunning
 - activemq::core::ActiveMQSessionExecutor, 462
 - activemq::core::MessageDispatchChannel, 2227
- isScheduled
 - decaf::util::TimerTask, 3200
- isServerAuthority
 - decaf::internal::net::URIType, 3343
- isShutdownInfo
 - activemq::commands::BaseCommand, 655
 - activemq::commands::Command, 1066
 - activemq::commands::ShutdownInfo, 2936
- isSizePrefixDisabled
 - activemq::commands::WireFormatInfo, 3375
 - activemq::wireformat::openwire::OpenWireFormat, 2453
- isSlaveBroker
 - activemq::commands::BrokerInfo, 780
- isStackTraceEnabled
 - activemq::commands::WireFormatInfo, 3375
 - activemq::wireformat::openwire::OpenWireFormat, 2454
- isStart
 - activemq::commands::ConsumerControl, 1253
- isStarted
 - activemq::core::ActiveMQConnection, 239
 - activemq::core::ActiveMQSession, 456
- isStop
 - activemq::commands::ConsumerControl, 1253
- isSuspend
 - activemq::commands::ConnectionControl, 1138
- isSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 268
- isTcpNoDelayEnabled
 - activemq::commands::WireFormatInfo, 3375
 - activemq::wireformat::openwire::OpenWireFormat, 2454
- isTemporary
 - activemq::commands::ActiveMQDestination, 281
- isTightEncodingEnabled
 - activemq::commands::WireFormatInfo, 3375
 - activemq::wireformat::openwire::OpenWireFormat, 2454
- isTopic
 - activemq::commands::ActiveMQDestination, 281
- isTrackMessages
 - activemq::state::ConnectionStateTracker, 1246
 - activemq::transport::failover::FailoverTransport, 1618
- isTrackTransactions
 - activemq::state::ConnectionStateTracker, 1246
- isTransacted
 - activemq::cmsutil::PooledSession, 2516
 - activemq::core::ActiveMQSession, 456
 - cms::Session, 2850
- isTransactionInfo
 - activemq::commands::BaseCommand, 655
 - activemq::commands::Command, 1066
 - activemq::commands::TransactionInfo, 3242
- isUpperCase
 - decaf::lang::Character, 986
- isUseAsyncSend
 - activemq::core::ActiveMQConnectionSupport, 257
- isUseExponentialBackOff
 - activemq::transport::failover::FailoverTransport, 1618
- isValid
 - activemq::commands::WireFormatInfo, 3375
 - decaf::internal::net::URIType, 3343
- isValidDomainName
 - decaf::internal::net::URIHelper, 3324
- isValidHexChar
 - decaf::internal::net::URIHelper, 3324
- isValidHost
 - decaf::internal::net::URIHelper, 3324
- isValidIP4Word
 - decaf::internal::net::URIHelper, 3324
- isValidIP6Address
 - decaf::internal::net::URIHelper, 3325
- isValidIPv4Address
 - decaf::internal::net::URIHelper, 3325
- isWaitingForResponse
 - activemq::state::Tracked, 3216
- isWhitespace
 - decaf::lang::Character, 986
- isWildcard

- activemq::commands::ActiveMQDestination, 281
- isWireFormatInfo
 - activemq::commands::BaseCommand, 655
 - activemq::commands::Command, 1066
 - activemq::commands::WireFormatInfo, 3375
- itemExists
 - activemq::commands::ActiveMQMapMessage, 316
 - cms::MapMessage, 2112
- iterate
 - activemq::core::ActiveMQConsumer, 269
 - activemq::core::ActiveMQSessionExecutor, 462
 - activemq::threads::CompositeTaskRunner, 1093
 - activemq::threads::Task, 3148
 - activemq::transport::failover::BackupTransportPool, 649
 - activemq::transport::failover::CloseTransportsTask, 1022
 - activemq::transport::failover::FailoverTransport, 1618
- iterator
 - decaf::lang::Iterable, 1830
 - decaf::util::concurrent::SynchronousQueue, 3141
 - decaf::util::PriorityQueue, 2575
 - decaf::util::StlList, 3025
 - decaf::util::StlQueue, 3046
 - decaf::util::StlSet, 3055
- JournalQueueAck
 - activemq::commands::JournalQueueAck, 1835
- JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 1851
 - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 1839
 - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 1847
 - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 1843
 - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 1855
- JournalTopicAck
 - activemq::commands::JournalTopicAck, 1859
- JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 1868
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 1864
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 1872
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 1876
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 1880
- activemq::commands::JournalTrace, 1884
- JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 1899
 - activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 1887
 - activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 1895
 - activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 1891
 - activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 1903
- JournalTransaction
 - activemq::commands::JournalTransaction, 1907
- JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1923
 - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1911
 - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1915
 - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1919
 - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1927
- KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 1931
- KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 1930
 - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 1934
 - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 1942
 - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 1946
 - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 1938
- KeyException
 - decaf::security::KeyException, 1955, 1956
- keySet
 - decaf::util::concurrent::StlMap, 1109

- decaf::util::Map, 2101
- decaf::util::StlMap, 3037
- lastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2706
- lastIndexOf
 - decaf::util::List, 1987
 - decaf::util::StlList, 3026
- lastMessageId
 - activemq::commands::MessageAck, 2193
- lastNakNumber
 - activemq::commands::ReplayCommand, 2755
- LastPartialCommand
 - activemq::commands::LastPartialCommand, 1959
- LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 1966
 - activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 1962
 - activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 1970
 - activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 1974
 - activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 1978
- length
 - decaf::lang::CharSequence, 1015
 - decaf::nio::CharBuffer, 1007
- Less
 - decaf::util::comparators::Less, 1981
- Level
 - decaf::util::logging, 143
- limit
 - decaf::nio::Buffer, 806, 807
- LineNumber
 - activemq::commands::BrokerError::StackTraceElement, 2993
- LIST_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- listener
 - activemq::transport::TransportFilter, 3288
- listIterator
 - decaf::util::List, 1987, 1988
 - decaf::util::StlList, 3026, 3027
- listValue
 - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2552
- load
 - decaf::util::Properties, 2660, 2662
- LocalTransactionId
 - activemq::commands::LocalTransactionId, 1994
- LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2010
 - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 1998
 - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2002
 - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2006
 - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2014
- Lock
 - decaf::util::concurrent::Lock, 2023
- lock
 - activemq::core::MessageDispatchChannel, 2227
 - decaf::internal::io::StandardErrorOutputStream, 2005
 - decaf::internal::io::StandardInputStream, 2002
 - decaf::internal::io::StandardOutputStream, 2009
 - decaf::internal::util::concurrent::MutexImpl, 2301
 - decaf::internal::util::concurrent::SynchronizableImpl, 2164
 - decaf::io::BlockingByteArrayInputStream, 726
 - decaf::io::ByteArrayInputStream, 894
 - decaf::io::ByteArrayOutputStream, 902
 - decaf::io::FilterInputStream, 1634
 - decaf::io::FilterOutputStream, 1642
 - decaf::net::SocketInputStream, 2979
 - decaf::net::SocketOutputStream, 2986
 - decaf::util::AbstractCollection, 154
 - decaf::util::concurrent::ConcurrentStlMap, 1110
 - decaf::util::concurrent::Lock, 2024
 - decaf::util::concurrent::locks::Lock, 2018
 - decaf::util::concurrent::locks::ReentrantLock, 2694
 - decaf::util::concurrent::Mutex, 2386
 - decaf::util::concurrent::Synchronizable, 3123
 - decaf::util::StlMap, 3037
 - decaf::util::StlQueue, 3046
- lock_count
 - decaf::util::concurrent::MutexHandle, 2390
- lock_owner
 - decaf::util::concurrent::MutexHandle, 2390
- lockInterruptibly
 - decaf::util::concurrent::locks::Lock, 2019
 - decaf::util::concurrent::locks::ReentrantLock, 2695

- log
 - decaf::util::logging::Logger, 2033, 2034
 - decaf::util::logging::LogWriter, 2055, 2056
 - decaf::util::logging::SimpleLogger, 2963
- LOGDECAF_DEBUG
 - LoggerDefines.h, 4164
- LOGDECAF_DEBUG_1
 - LoggerDefines.h, 4164
- LOGDECAF_DECLARE
 - LoggerDefines.h, 4164
- LOGDECAF_DECLARE_LOCAL
 - LoggerDefines.h, 4165
- LOGDECAF_ERROR
 - LoggerDefines.h, 4165
- LOGDECAF_FATAL
 - LoggerDefines.h, 4165
- LOGDECAF_INFO
 - LoggerDefines.h, 4165
- LOGDECAF_INITIALIZE
 - LoggerDefines.h, 4165
- LOGDECAF_WARN
 - LoggerDefines.h, 4165
- Logger
 - decaf::util::logging::Logger, 2029
- LoggerDefines.h
 - LOGDECAF_DEBUG, 4164
 - LOGDECAF_DEBUG_1, 4164
 - LOGDECAF_DECLARE, 4164
 - LOGDECAF_DECLARE_LOCAL, 4165
 - LOGDECAF_ERROR, 4165
 - LOGDECAF_FATAL, 4165
 - LOGDECAF_INFO, 4165
 - LOGDECAF_INITIALIZE, 4165
 - LOGDECAF_WARN, 4165
- LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 2037
- LoggingInputStream
 - activemq::io::LoggingInputStream, 2038
- LoggingOutputStream
 - activemq::io::LoggingOutputStream, 2040
- LoggingTransport
 - activemq::transport::logging::LoggingTransport, 2042
- LogManager
 - decaf::util::logging::LogManager, 2047
- LogRecord
 - decaf::util::logging::LogRecord, 2051
- LogWriter
 - decaf::util::logging::LogWriter, 2055
- Long
 - decaf::lang::Long, 2059
- LONG_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- LongArrayBuffer
 - decaf::internal::nio::LongArrayBuffer, 2072, 2073
- longBitsToDouble
 - decaf::lang::Double, 1543
- LongBuffer
 - decaf::nio::LongBuffer, 2081
- LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 2090
- longValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2552
 - decaf::lang::Byte, 844
 - decaf::lang::Character, 987
 - decaf::lang::Double, 1543
 - decaf::lang::Float, 1653
 - decaf::lang::Integer, 1768
 - decaf::lang::Long, 2062
 - decaf::lang::Number, 2433
 - decaf::lang::Short, 2910
 - decaf::util::concurrent::atomic::AtomicInteger, 637
- lookup
 - decaf::security_-
provider::SecurityProviderMap, 2824
- looseMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMar
698
 - activemq::wireformat::openwire::marshal::DataStreamMarshal
1429
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM
182
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesI
218
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestin
290
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapM
327
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessa
350
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObject
391
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue
428
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStream
484
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempI
508
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp
533
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp5
558
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextM
583

activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	2244	activemq::wireformat::openwire::marshal::v1::MessageDispatcher
607		
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	2270	activemq::wireformat::openwire::marshal::v1::MessageDispatcher
665		
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2301	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller
760		
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2326	activemq::wireformat::openwire::marshal::v1::MessageMarshaller
788		
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2360	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller
1145		
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2410	activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilter
1169		
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2490	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller
1196		
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2600	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller
1226		
activemq::wireformat::openwire::marshal::v1::ConsumerGroupViewMarshaller	2624	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller
1264		
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2649	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller
1289		
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	2708	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller
1314		
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	2741	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller
1343		
activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller	2773	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
1368		
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2807	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller
1403		
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	2862	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller
1502		
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	2882	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller
1532		
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	2938	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller
1590		
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3103	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller
1684		
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3221	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller
1785		
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3246	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller
1851		
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	3396	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller
1868		
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	3420	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller
1899		
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	194	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller
1923		
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	230	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller
1950		
activemq::wireformat::openwire::marshal::v1::LastReceivedCommandMarshaller	302	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
1966		
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	339	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller
2010		
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	362	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
2211		

activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	1864	activemq::wireformat::openwire::marshal::v2::JournalTopicAck
403		
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller	1887	activemq::wireformat::openwire::marshal::v2::JournalTraceMa
440		
activemq::wireformat::openwire::marshal::v2::ActiveMQStackedMessageMarshaller	1911	activemq::wireformat::openwire::marshal::v2::JournalTransact
496		
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDeflationMarshaller	1934	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoM
520		
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	1962	activemq::wireformat::openwire::marshal::v2::LastPartialComr
545		
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	1998	activemq::wireformat::openwire::marshal::v2::LocalTransaction
570		
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2195	activemq::wireformat::openwire::marshal::v2::MessageAckMar
595		
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	2232	activemq::wireformat::openwire::marshal::v2::MessageDispat ch
619		
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2258	activemq::wireformat::openwire::marshal::v2::MessageDispat ch
686		
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2285	activemq::wireformat::openwire::marshal::v2::MessageIdMarsh
772		
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2306	activemq::wireformat::openwire::marshal::v2::MessageMarshal
800		
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2352	activemq::wireformat::openwire::marshal::v2::MessagePullMar
1157		
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2398	activemq::wireformat::openwire::marshal::v2::NetworkBridgeF
1181		
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2478	activemq::wireformat::openwire::marshal::v2::PartialCommand
1208		
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2584	activemq::wireformat::openwire::marshal::v2::ProducerAckMar
1234		
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2612	activemq::wireformat::openwire::marshal::v2::ProducerIdMars
1272		
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2637	activemq::wireformat::openwire::marshal::v2::ProducerInfoMa
1297		
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	2712	activemq::wireformat::openwire::marshal::v2::RemoveInfoMars
1326		
activemq::wireformat::openwire::marshal::v2::ContentCommandMarshaller	2737	activemq::wireformat::openwire::marshal::v2::RemoveSubscrip
1351		
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	2757	activemq::wireformat::openwire::marshal::v2::ReplayCommand
1376		
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2792	activemq::wireformat::openwire::marshal::v2::ResponseMarsha
1415		
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	2866	activemq::wireformat::openwire::marshal::v2::SessionIdMarsha
1490		
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	2890	activemq::wireformat::openwire::marshal::v2::SessionInfoMars
1516		
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	2950	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMa
1586		
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3119	activemq::wireformat::openwire::marshal::v2::SubscriptionInfo
1680		
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	3225	activemq::wireformat::openwire::marshal::v2::TransactionIdMa
1781		
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3262	activemq::wireformat::openwire::marshal::v2::TransactionInfo
1839		

activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	openwire::marshal::v3::DataResponseM
3388	1399
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	openwire::marshal::v3::DestinationInfoM
3428	1494
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobWireFormatMessageMarshaller	openwire::marshal::v3::DiscoveryEventM
178	1520
activemq::wireformat::openwire::marshal::v3::ActiveMQByteWireFormatMessageMarshaller	openwire::marshal::v3::ExceptionRespon
214	1594
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationFormatMarshaller	openwire::marshal::v3::FlushCommandL
286	1688
activemq::wireformat::openwire::marshal::v3::ActiveMQMapWireFormatMessageMarshaller	openwire::marshal::v3::IntegerResponse
323	1789
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	openwire::marshal::v3::JournalQueueAc
346	1847
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectWireFormatMessageMarshaller	openwire::marshal::v3::JournalTopicAck
387	1872
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMarshaller	openwire::marshal::v3::JournalTraceMa
424	1895
activemq::wireformat::openwire::marshal::v3::ActiveMQSequenceWireFormatMessageMarshaller	openwire::marshal::v3::JournalTransact
480	1915
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	openwire::marshal::v3::KeepAliveInfoM
504	1942
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	openwire::marshal::v3::LastPartialComm
529	1970
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	openwire::marshal::v3::LocalTransaction
554	2002
activemq::wireformat::openwire::marshal::v3::ActiveMQTextWireFormatMessageMarshaller	openwire::marshal::v3::MessageAckMar
579	2203
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	openwire::marshal::v3::MessageDispatc
603	2240
activemq::wireformat::openwire::marshal::v3::BaseCommandWireFormatMarshaller	openwire::marshal::v3::MessageDispatc
658	2266
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	openwire::marshal::v3::MessageIdMarsh
756	2293
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	openwire::marshal::v3::MessageMarshal
784	2316
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	openwire::marshal::v3::MessagePullMar
1141	2356
activemq::wireformat::openwire::marshal::v3::ConnectionFromMarshaller	openwire::marshal::v3::NetworkBridgeF
1165	2402
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	openwire::marshal::v3::PartialCommand
1192	2482
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	openwire::marshal::v3::ProducerAckMar
1218	2588
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	openwire::marshal::v3::ProducerIdMars
1256	2616
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	openwire::marshal::v3::ProducerInfoMa
1281	2653
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	openwire::marshal::v3::RemoveInfoMars
1310	2716
activemq::wireformat::openwire::marshal::v3::ConnectionFactoryMarshaller	openwire::marshal::v3::RemoveSubscrip
1335	2745
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	openwire::marshal::v3::ReplayCommand
1360	2761

activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	1200
2797	
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	1222
2870	
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	1260
2898	
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	1285
2946	
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	1318
3111	
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	1339
3237	
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	1364
3254	
activemq::wireformat::openwire::marshal::v3::WireFormatMarshaller	1411
3380	
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1498
3416	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	1524
186	
activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	1598
222	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1692
294	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	1793
331	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	1843
354	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	1876
395	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	1891
432	
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	1919
488	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	1946
512	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	1974
537	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2006
562	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2207
587	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2236
611	
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	2262
672	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2289
764	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2311
792	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2368
1149	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2414
1173	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::ConsumerController	
activemq::wireformat::openwire::marshal::v4::ControlCommand	
activemq::wireformat::openwire::marshal::v4::DataArrayResponse	
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::DiscoveryEventManager	
activemq::wireformat::openwire::marshal::v4::ExceptionResponse	
activemq::wireformat::openwire::marshal::v4::FlushCommand	
activemq::wireformat::openwire::marshal::v4::IntegerResponse	
activemq::wireformat::openwire::marshal::v4::JournalQueueAck	
activemq::wireformat::openwire::marshal::v4::JournalTopicAck	
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTransaction	
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::LastPartialCommand	
activemq::wireformat::openwire::marshal::v4::LocalTransaction	
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFactory	

activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	openwire::marshal::v5::ActiveMQTextM
2486	591
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::ActiveMQTopicC
2592	615
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::BaseCommandM
2628	679
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::BrokerIdMarsha
2645	768
activemq::wireformat::openwire::marshal::v4::RemoteInfoMarshaller	openwire::marshal::v5::BrokerInfoMarsh
2720	796
activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	openwire::marshal::v5::ConnectionCont
2749	1153
activemq::wireformat::openwire::marshal::v4::ReplaceCommandMarshaller	openwire::marshal::v5::ConnectionError
2769	1177
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	openwire::marshal::v5::ConnectionIdMa
2812	1204
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	openwire::marshal::v5::ConnectionInfoM
2874	1230
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	openwire::marshal::v5::ConsumerContro
2886	1268
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	openwire::marshal::v5::ConsumerIdMar
2942	1293
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	openwire::marshal::v5::ConsumerInfoMa
3107	1322
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	openwire::marshal::v5::ControlComman
3229	1347
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	openwire::marshal::v5::DataArrayRespo
3250	1372
activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller	openwire::marshal::v5::DataResponseM
3392	1407
activemq::wireformat::openwire::marshal::v4::XATransactionMarshaller	openwire::marshal::v5::DestinationInfoM
3424	1506
activemq::wireformat::openwire::marshal::v5::ActiveMQBinaryMessageMarshaller	openwire::marshal::v5::DiscoveryEventM
190	1528
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	openwire::marshal::v5::ExceptionRespor
226	1602
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	openwire::marshal::v5::FlushCommandL
298	1696
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	openwire::marshal::v5::IntegerResponse
335	1797
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	openwire::marshal::v5::JournalQueueAc
358	1855
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	openwire::marshal::v5::JournalTopicAck
399	1880
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	openwire::marshal::v5::JournalTraceMa
436	1903
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	openwire::marshal::v5::JournalTransact
492	1927
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	openwire::marshal::v5::KeepAliveInfoM
516	1938
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	openwire::marshal::v5::LastPartialComr
541	1978
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	openwire::marshal::v5::LocalTransaction
566	2014

- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2199
- looseMarshalNestedObject 699
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2248
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2274
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2297
- looseMarshalObjectArray 2297
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2321
- looseMarshalString 2364
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2364
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2406
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2494
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2596
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2620
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2641
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2724
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2733
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2765
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2802
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 2858
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2894
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2954
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3115
- activemq::wireformat::openwire::marshal::v5::TransactionMarshaller, 3233
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3258
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3384
- activemq::wireformat::openwire::marshal::v5::XATransactionMarshaller, 3432
- looseMarshalBrokerError 698
- activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller, 698
- looseMarshalCachedObject 699
- activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller, 699
- looseMarshalLong 1169
- activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller, 1169
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller, 1433
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller, 182
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 218
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationInfoMarshaller, 290
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller, 327
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 350
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller, 391
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 428
- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller, 484
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller, 508
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateInfoMarshaller, 533
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateInfoMarshaller, 558
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller, 583
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 607
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 666
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 769
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 788
- activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller, 1145
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1169

activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2491
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	1196
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	1226
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2600
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	1264
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2624
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	1289
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	2649
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	1314
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	2708
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	1343
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	2741
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	1369
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	2773
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	1404
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	2808
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	1502
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	2862
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	1532
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	2882
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	1591
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	2938
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	1684
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3103
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	1786
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3221
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	1851
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3246
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	1868
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3396
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	1899
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	3420
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	1923
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	194
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	1950
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	230
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	1967
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	302
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	2010
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	339
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	2211
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	362
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	2244
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	403
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2270
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	440
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	2301
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	496
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	2326
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	520
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	2360
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	545
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	2410
activemq::wireformat::openwire::marshal::v2::ActiveMQTempMarshaller	570

activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2195
595	
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMessageMarshaller	2232
619	
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	2258
687	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	2285
772	
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2306
800	
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2352
1157	
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2398
1181	
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2479
1208	
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2584
1234	
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	2612
1272	
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2637
1297	
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	2712
1326	
activemq::wireformat::openwire::marshal::v2::ContactCommandMarshaller	2737
1351	
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2757
1377	
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2793
1416	
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	2866
1490	
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	2890
1516	
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	2950
1587	
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	3119
1680	
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	3225
1782	
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3262
1839	
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	3388
1864	
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	3428
1887	
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	178
1911	
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	214
1934	
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	286
1963	
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	323
1998	

activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	1847	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	1847
346		activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	1872
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1872	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	1872
387		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1895
424		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1895
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1915	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	1915
480		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1915
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1942	activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	1942
504		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1942
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1971	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	1971
529		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	1971
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2002	activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller	2002
554		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2002
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2203	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2203
579		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2203
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2240	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2240
603		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2240
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2266	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2266
659		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2266
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2293	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2293
756		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2293
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2316	activemq::wireformat::openwire::marshal::v3::MessageMarshaller	2316
784		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2316
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2356	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2356
1141		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2356
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2402	activemq::wireformat::openwire::marshal::v3::NetworkBridgeMarshaller	2402
1165		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2402
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2483	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2483
1192		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2483
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2588	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	2588
1218		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2588
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2616	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	2616
1256		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2616
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2653	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	2653
1281		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2653
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2716	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	2716
1310		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2716
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2745	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	2745
1335		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2745
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2761	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	2761
1361		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2761
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2798	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	2798
1400		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2798
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2870	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	2870
1494		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2870
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2898	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	2898
1520		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2898
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2946	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	2946
1595		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	2946
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	3111	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	3111
1688		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	3111
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	3237	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3237
1790		activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	3237

activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	1365
3254	
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	1412
3380	
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1498
3416	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireFormatMarshaller	1524
186	
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	1599
222	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1692
294	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	1794
331	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	1843
354	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	1876
395	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMarshaller	1891
432	
activemq::wireformat::openwire::marshal::v4::ActiveMQStackWireFormatMarshaller	1919
488	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	1946
512	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	1975
537	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2006
562	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextWireFormatMarshaller	2207
587	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2236
611	
activemq::wireformat::openwire::marshal::v4::BaseCommandWireFormatMarshaller	2262
673	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2289
764	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2311
792	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2368
1149	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2414
1173	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	2487
1200	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	2592
1222	
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	2628
1260	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	2645
1285	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	2720
1318	
activemq::wireformat::openwire::marshal::v4::ContentCommandMarshaller	2749
1339	

activemq::wireformat::openwire::marshal::v4::ReplyCommandMarshaller	openwire::marshal::v5::ConnectionError
2769	1177
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	openwire::marshal::v5::ConnectionIdMar
2813	1204
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	openwire::marshal::v5::ConnectionInfoM
2874	1230
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	openwire::marshal::v5::ConsumerContro
2886	1268
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	openwire::marshal::v5::ConsumerIdMar
2942	1293
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	openwire::marshal::v5::ConsumerInfoMa
3107	1322
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	openwire::marshal::v5::ControlComm
3229	1347
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	openwire::marshal::v5::DataArrayRespo
3250	1373
activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller	openwire::marshal::v5::DataResponseM
3392	1408
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	openwire::marshal::v5::DestinationInfoM
3424	1506
activemq::wireformat::openwire::marshal::v5::ActiveMQBlockWireFormatMarshaller	openwire::marshal::v5::DiscoveryEventM
190	1528
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireFormatMarshaller	openwire::marshal::v5::ExceptionRespo
226	1603
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	openwire::marshal::v5::FlushCommandL
298	1696
activemq::wireformat::openwire::marshal::v5::ActiveMQMapWireFormatMarshaller	openwire::marshal::v5::IntegerResponse
335	1798
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	openwire::marshal::v5::JournalQueueAc
358	1855
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireFormatMarshaller	openwire::marshal::v5::JournalTopicAck
399	1880
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueWireFormatMarshaller	openwire::marshal::v5::JournalTraceMa
436	1903
activemq::wireformat::openwire::marshal::v5::ActiveMQStackingMessageMarshaller	openwire::marshal::v5::JournalTransact
492	1927
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	openwire::marshal::v5::KeepAliveInfoM
516	1938
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	openwire::marshal::v5::LastPartialCom
541	1979
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	openwire::marshal::v5::LocalTransaction
566	2014
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	openwire::marshal::v5::MessageAckMar
591	2199
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	openwire::marshal::v5::MessageDispatc
615	2248
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	openwire::marshal::v5::MessageDispatc
680	2274
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	openwire::marshal::v5::MessageIdMarsh
768	2297
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	openwire::marshal::v5::MessageMarshal
796	2321
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	openwire::marshal::v5::MessagePullMar
1153	2364

- activemq::wireformat::stomp::StompWireFormatMessageAckMarshaller
 - 3073
- activemq::wireformat::WireFormat, 3365
- marshalledProperties
 - activemq::commands::Message, 2161
- marshalledSize
 - activemq::wireformat::openwire::utils::BooleanStream, 2203
 - 740
- marshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2548
- marshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2548
- marshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2548
- masterBroker
 - activemq::commands::BrokerInfo, 782
- Math
 - decaf::lang::Math, 2128
- max
 - decaf::lang::Math, 2130, 2131
- MAX_RADIX
 - decaf::lang::Character, 989
- MAX_VALUE
 - decaf::lang::Byte, 848
 - decaf::lang::Character, 989
 - decaf::lang::Double, 1547
 - decaf::lang::Float, 1657
 - decaf::lang::Integer, 1775
 - decaf::lang::Long, 2070
 - decaf::lang::Short, 2914
- maximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1307
- MemoryUsage
 - activemq::util::MemoryUsage, 2142
- MESSAGE
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- Message
 - activemq::commands::Message, 2149
- message
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2691
 - activemq::commands::JournalTrace, 1885
 - activemq::commands::MessageDispatch, 2223
 - decaf::lang::Exception, 1580
- MessageAck
 - activemq::commands::MessageAck, 2189
- messageAck
 - activemq::commands::JournalQueueAck, 1837
- MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2211
 - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2195
 - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2207
 - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2199
 - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2199
- messageCount
 - activemq::commands::MessageAck, 2193
- MessageDispatch
 - activemq::commands::MessageDispatch, 2220
- MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 2225
- MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2244
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2232
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2240
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2236
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2248
- MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 2252
- MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2270
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2258
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2266
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2262
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2274
- MessageEOFException
 - cms::MessageEOFException, 2277
- MessageFormatException
 - cms::MessageFormatException, 2278
- MessageId
 - activemq::commands::MessageId, 2280
- messageId
 - activemq::commands::JournalTopicAck, 1862
 - activemq::commands::Message, 2161

- activemq::commands::MessageDispatchNotification, 3213
- 2256
- min
- activemq::commands::MessagePull, 2350
- decaf::lang::Math, 2132, 2133
- MessageIdMarshaller
- MIN_RADIX
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 989
- 2301
- MIN_VALUE
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 848
- 2285
- decaf::lang::Character, 989
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 1547
- 2293
- decaf::lang::Float, 1657
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 1775
- 2289
- decaf::lang::Long, 2070
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2914
- 2297
- MINUTES
- MessageMarshaller
- decaf::util::concurrent::TimeUnit, 3213
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2326
- activemq::transport::mock::MockTransport, 2378
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2306
- Mutex
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2316
- decaf::util::concurrent::Mutex, 2386
- mutex
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2311
- decaf::io::FilterOutputStream, 1646
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2321
- decaf::util::concurrent::ConditionHandle, 1124
- MessageNotReadableException
- cms::MessageNotReadableException, 2330
- decaf::util::concurrent::MutexHandle, 2390
- MessageNotWriteableException
- MutexHandle
- cms::MessageNotWriteableException, 2331
- decaf::util::concurrent::MutexHandle, 2390
- MessagePropertyInterceptor
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2340
- decaf::util::UUID, 3359
- MessagePull
- NaN
- activemq::commands::MessagePull, 2347
- decaf::lang::Double, 1547
- MessagePullMarshaller
- decaf::lang::Float, 1657
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2360
- decaf::util::concurrent::TimeUnit, 3213
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2352
- decaf::lang::System, 3146
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2356
- activemq::transport::failover::FailoverTransport, 1618
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2368
- activemq::transport::IOTransport, 1826
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2364
- activemq::transport::mock::MockTransport, 2376
- messageSequenceId
- activemq::transport::Transport, 3275
- activemq::commands::JournalTopicAck, 1862
- activemq::transport::TransportFilter, 3285
- NEGATIVE_INFINITY
- MethodName
- decaf::lang::Double, 1547
- activemq::commands::BrokerError::StackTraceElement, 2993
- decaf::lang::Float, 1657
- NetworkBridgeFilter
- activemq::commands::NetworkBridgeFilter, 2394
- MICROSECONDS
- decaf::util::concurrent::TimeUnit, 3213
- MILLISECONDS
- NetworkBridgeFilterMarshaller

- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2410
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2398
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2402
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2414
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2406
- networkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2396
- networkConnection
 - activemq::commands::BrokerInfo, 782
- networkConsumerPath
 - activemq::commands::ConsumerInfo, 1307
- networkProperties
 - activemq::commands::BrokerInfo, 782
- networkSubscription
 - activemq::commands::ConsumerInfo, 1307
- networkTTL
 - activemq::commands::NetworkBridgeFilter, 2396
- newCondition
 - decaf::util::concurrent::locks::Lock, 2019
 - decaf::util::concurrent::locks::ReentrantLock, 2695
- newThread
 - decaf::util::concurrent::ThreadFactory, 3172
- next
 - activemq::transport::TransportFilter, 3288
 - decaf::util::Iterator, 1832
 - decaf::util::Random, 2678
- nextBoolean
 - decaf::util::Random, 2679
- nextBytes
 - decaf::util::Random, 2679
- nextDouble
 - decaf::util::Random, 2679
- nextFloat
 - decaf::util::Random, 2679
- nextGaussian
 - decaf::util::Random, 2679
- nextIndex
 - decaf::util::ListIterator, 1991
- nextInt
 - decaf::util::Random, 2680
- nextLong
 - decaf::util::Random, 2680
- nextToken
 - decaf::util::StringTokenizer, 3095
- node
 - decaf::net::URI, 3315
 - NoRouteToHostException
 - decaf::net::NoRouteToHostException, 2417, 2418
 - NoSuchAlgorithmException
 - decaf::security::NoSuchAlgorithmException, 2420, 2421
 - NoSuchElementException
 - decaf::lang::exceptions::NoSuchElementException, 2423, 2424
 - NoSuchProviderException
 - decaf::security::NoSuchProviderException, 2426, 2427
 - notify
 - activemq::core::MessageDispatchChannel, 2227
 - decaf::internal::io::StandardErrorOutputStream, 2996
 - decaf::internal::io::StandardInputStream, 3002
 - decaf::internal::io::StandardOutputStream, 3009
 - decaf::internal::util::concurrent::ConditionImpl, 1126
 - decaf::internal::util::concurrent::SynchronizableImpl, 3134
 - decaf::io::BlockingByteArrayInputStream, 727
 - decaf::io::ByteArrayInputStream, 895
 - decaf::io::ByteArrayOutputStream, 902
 - decaf::io::FilterInputStream, 1634
 - decaf::io::FilterOutputStream, 1643
 - decaf::net::SocketInputStream, 2980
 - decaf::net::SocketOutputStream, 2986
 - decaf::util::AbstractCollection, 154
 - decaf::util::concurrent::ConcurrentStlMap, 1110
 - decaf::util::concurrent::Mutex, 2386
 - decaf::util::concurrent::Synchronizable, 3124
 - decaf::util::StlMap, 3037
 - decaf::util::StlQueue, 3047
 - normalize

- activemq::core::MessageDispatchChannel, 227
- decaf::internal::io::StandardErrorOutputStream, 2996
- decaf::internal::io::StandardInputStream, 3003
- decaf::internal::io::StandardOutputStream, 3010
- decaf::internal::util::concurrent::ConditionImpl, 1127
- decaf::internal::util::concurrent::SynchronizableImpl, 3134
- decaf::io::BlockingByteArrayInputStream, 727
- decaf::io::ByteArrayInputStream, 895
- decaf::io::ByteArrayOutputStream, 903
- decaf::io::FilterInputStream, 1635
- decaf::io::FilterOutputStream, 1643
- decaf::net::SocketInputStream, 2980
- decaf::net::SocketOutputStream, 2986
- decaf::util::AbstractCollection, 154
- decaf::util::concurrent::ConcurrentStlMap, 1110
- decaf::util::concurrent::Mutex, 2386
- decaf::util::concurrent::Synchronizable, 3125
- decaf::util::StlMap, 3038
- decaf::util::StlQueue, 3047
- Null
 - decaf::util::logging, 144
- NULL_TYPE
 - activemq::util::PrimitiveValueNode, 2558
 - activemq::wireformat::openwire::OpenWireFormat, 2459
- NullPointerException
 - decaf::lang::exceptions::NullPointerException, 2429, 2430
- NUM_OPTIONS
 - activemq::core::ActiveMQConstants, 261
- NUM_PARAMS
 - activemq::core::ActiveMQConstants, 262
- NumberFormatException
 - decaf::lang::exceptions::NumberFormatException, 2435, 2436
- numberOfLeadingZeros
 - decaf::lang::Integer, 1768
 - decaf::lang::Long, 2063
- numberOfTrailingZeros
 - decaf::lang::Integer, 1768
 - decaf::lang::Long, 2063
- numWaiting
 - decaf::util::concurrent::ConditionHandle, 1124
- numWake
- decaf::util::concurrent::ConditionHandle, 1124
- objectId
 - activemq::commands::RemoveInfo, 2706
- Off
 - decaf::util::logging, 144
- offer
 - decaf::util::concurrent::SynchronousQueue, 3141
 - decaf::util::PriorityQueue, 2575
 - decaf::util::Queue, 2672
- offset
 - decaf::internal::nio::CharArrayBuffer, 997
- onCommand
 - activemq::core::ActiveMQConnection, 240
 - activemq::transport::correlator::ResponseCorrelator, 2788
 - activemq::transport::DefaultTransportListener, 1475
 - activemq::transport::failover::FailoverTransportListener, 1628
 - activemq::transport::inactivity::InactivityMonitor, 1734
 - activemq::transport::logging::LoggingTransport, 2043
 - activemq::transport::mock::InternalCommandListener, 1800
 - activemq::transport::TransportFilter, 3285
 - activemq::transport::TransportListener, 3289
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2463
- oneway
 - activemq::core::ActiveMQConnection, 240
 - activemq::core::ActiveMQSession, 456
 - activemq::transport::correlator::ResponseCorrelator, 2788
 - activemq::transport::failover::FailoverTransport, 1619
 - activemq::transport::inactivity::InactivityMonitor, 1735
 - activemq::transport::IOTransport, 1826
 - activemq::transport::logging::LoggingTransport, 2043
 - activemq::transport::mock::MockTransport, 2376
 - activemq::transport::Transport, 3276
 - activemq::transport::TransportFilter, 3285
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2463
- onException
 - activemq::core::ActiveMQConnection, 240

- activemq::transport::DefaultTransportListener, 1475
- activemq::transport::failover::BackupTransportListener, 645
- activemq::transport::failover::FailoverTransportListener, 1628
- activemq::transport::inactivity::InactivityMonitor, 1735
- activemq::transport::TransportFilter, 3285
- activemq::transport::TransportListener, 3290
- cms::ExceptionListener, 1581
- onMessage
 - cms::MessageListener, 2304
- onProducerAck
 - activemq::core::ActiveMQProducer, 410
- onPropertyChanged
 - decaf::util::logging::PropertiesChangeListener, 2666
- onResponse
 - activemq::state::Tracked, 3216
- onSend
 - activemq::commands::ActiveMQBytesMessage, 202
 - activemq::commands::ActiveMQMessageTemplate, 375
 - activemq::commands::ActiveMQStreamMessage, 468
 - activemq::commands::Message, 2156
- onTaskComplete
 - decaf::util::concurrent::TaskListener, 3149
- onTaskCompleted
 - decaf::util::concurrent::PooledThreadListener, 2520
 - decaf::util::concurrent::ThreadPool, 3178
- onTaskException
 - decaf::util::concurrent::PooledThreadListener, 2521
 - decaf::util::concurrent::TaskListener, 3149
 - decaf::util::concurrent::ThreadPool, 3178
- onTaskStarted
 - decaf::util::concurrent::PooledThreadListener, 2521
 - decaf::util::concurrent::ThreadPool, 3179
- onTransportException
 - activemq::transport::correlator::ResponseCorrelator, 2789
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2464
- OpenSSLX500Principal
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX500Principal, 2439
- OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2451
 - OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2460
 - OpenWireFormatNegotiator
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2462
 - OpenWireResponseBuilder
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2466
 - OpenwireStringSupport
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2468
 - operationType
 - activemq::commands::DestinationInfo, 1488
 - activemq::commands::BrokerId, 753
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::MessageId, 2282
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionId, 2855
 - activemq::commands::TransactionId, 3219
 - activemq::commands::XATransactionId, 3413
 - decaf::lang::Boolean, 734
 - decaf::lang::Byte, 844
 - decaf::lang::Character, 987
 - decaf::lang::Comparable, 1084
 - decaf::lang::Double, 1543, 1544
 - decaf::lang::Float, 1653, 1654
 - decaf::lang::Integer, 1769
 - decaf::lang::Long, 2064
 - decaf::lang::Short, 2910
 - decaf::net::URI, 3315
 - decaf::nio::ByteBuffer, 928
 - decaf::nio::CharBuffer, 1007
 - decaf::nio::DoubleBuffer, 1562
 - decaf::nio::FloatBuffer, 1671
 - decaf::nio::IntBuffer, 1757
 - decaf::nio::LongBuffer, 2085
 - decaf::nio::ShortBuffer, 2929
 - decaf::util::concurrent::TimeUnit, 3208
 - decaf::util::Date, 1469
 - decaf::util::UUID, 3360
 - operator*
 - decaf::lang::Pointer, 2501
 - operator()
 - decaf::lang::PointerComparator, 2504
 - decaf::util::Comparator, 1087

- decaf::util::comparators::Less, 1982
- std::less< decaf::lang::Pointer< T > >, 1983
- operator->
 - decaf::lang::Pointer, 2501
- operator=
 - activemq::commands::BrokerId, 753
 - activemq::commands::BrokerInfo, 780
 - activemq::commands::ConnectionControl, 1138
 - activemq::commands::ConnectionError, 1162
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConnectionInfo, 1215
 - activemq::commands::ConsumerControl, 1253
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::ConsumerInfo, 1305
 - activemq::commands::ControlCommand, 1332
 - activemq::commands::DataArrayResponse, 1358
 - activemq::commands::DataResponse, 1397
 - activemq::commands::DestinationInfo, 1487
 - activemq::commands::DiscoveryEvent, 1513
 - activemq::commands::ExceptionResponse, 1584
 - activemq::commands::FlushCommand, 1678
 - activemq::commands::IntegerResponse, 1779
 - activemq::commands::JournalQueueAck, 1836
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::JournalTrace, 1885
 - activemq::commands::JournalTransaction, 1908
 - activemq::commands::KeepAliveInfo, 1932
 - activemq::commands::LastPartialCommand, 1960
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::Message, 2156
 - activemq::commands::MessageAck, 2191
 - activemq::commands::MessageDispatch, 2222
 - activemq::commands::MessageDispatchNotification, 2254
 - activemq::commands::MessageId, 2282
 - activemq::commands::MessagePull, 2349
 - activemq::commands::NetworkBridgeFilter, 2395
 - activemq::commands::PartialCommand, 2475
 - activemq::commands::ProducerAck, 2581
 - activemq::commands::ProducerId, 2609
 - activemq::commands::ProducerInfo, 2634
 - activemq::commands::RemoveInfo, 2705
 - activemq::commands::RemoveSubscriptionInfo, 2730
 - activemq::commands::ReplayCommand, 2754
 - activemq::commands::Response, 2783
 - activemq::commands::SessionId, 2856
 - activemq::commands::SessionInfo, 2879
 - activemq::commands::ShutdownInfo, 2936
 - activemq::commands::SubscriptionInfo, 3100
 - activemq::commands::TransactionId, 3219
 - activemq::commands::TransactionInfo, 3242
 - activemq::commands::XATransactionId, 3413
- activemq::library::ActiveMQCPP, 273
- activemq::util::PrimitiveValueNode, 2565
- decaf::lang::Exception, 1578
- decaf::lang::Pointer, 2502
- decaf::util::AbstractCollection, 154
- decaf::util::Date, 1469
- decaf::util::logging::LogManager, 2049
- decaf::util::PriorityQueue, 2575, 2576
- decaf::util::Properties, 2662
- operator==
 - activemq::commands::BrokerId, 753
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::MessageId, 2282
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionId, 2856
 - activemq::commands::TransactionId, 3219
 - activemq::commands::XATransactionId, 3413
- activemq::util::PrimitiveValueNode, 2565
- decaf::lang, 125
- decaf::lang::Boolean, 735
- decaf::lang::Byte, 845
- decaf::lang::Character, 987, 988
- decaf::lang::Comparable, 1085
- decaf::lang::Double, 1544
- decaf::lang::Float, 1654
- decaf::lang::Integer, 1769, 1770
- decaf::lang::Long, 2064

- decaf::lang::Pointer, 2502, 2503
- decaf::lang::Short, 2911
- decaf::net::URI, 3316
- decaf::nio::ByteBuffer, 928
- decaf::nio::CharBuffer, 1007
- decaf::nio::DoubleBuffer, 1563
- decaf::nio::FloatBuffer, 1671
- decaf::nio::IntBuffer, 1757
- decaf::nio::LongBuffer, 2086
- decaf::nio::ShortBuffer, 2930
- decaf::util::concurrent::TimeUnit, 3208
- decaf::util::Date, 1470
- decaf::util::UUID, 3360
- operator[]
 - activemq::wireformat::openwire::utils::HexTable, 1715
 - decaf::internal::util::ByteArrayAdapter, 863
- optimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1307
- options
 - activemq::commands::ActiveMQDestination, 284
- ordered
 - activemq::commands::ActiveMQDestination, 284
- orderedTarget
 - activemq::commands::ActiveMQDestination, 284
- originalDestination
 - activemq::commands::Message, 2161
- originalTransactionId
 - activemq::commands::Message, 2161
- outputStream
 - decaf::io::FilterOutputStream, 1646
- own
 - decaf::io::FilterInputStream, 1639
 - decaf::io::FilterOutputStream, 1646
- PARAM_CLIENTID
 - activemq::core::ActiveMQConstants, 262
- PARAM_PASSWORD
 - activemq::core::ActiveMQConstants, 262
- PARAM_USERNAME
 - activemq::core::ActiveMQConstants, 262
- parent
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2605
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2691
- park
 - decaf::util::concurrent::locks::LockSupport, 2026
- parkNanos
 - decaf::util::concurrent::locks::LockSupport, 2026
- decaf::util::concurrent::locks::LockSupport, 2026
- parkUntil
 - decaf::util::concurrent::locks::LockSupport, 2027
- parse
 - decaf::internal::util::HexStringParser, 1713
- parseAuthority
 - decaf::internal::net::URIHelper, 3325
- parseBoolean
 - decaf::lang::Boolean, 735
- parseByte
 - decaf::lang::Byte, 845, 846
- parseComposite
 - activemq::util::URISupport, 3332
- parseDouble
 - decaf::internal::util::HexStringParser, 1713
 - decaf::lang::Double, 1545
- parseFloat
 - decaf::internal::util::HexStringParser, 1714
 - decaf::lang::Float, 1655
- parseInt
 - decaf::lang::Integer, 1770
- parseLong
 - decaf::lang::Long, 2065
- parseQuery
 - activemq::util::URISupport, 3333
- parseServerAuthority
 - decaf::net::URI, 3316
- parseShort
 - decaf::lang::Short, 2911
- parseURI
 - decaf::internal::net::URIHelper, 3326
- parseURL
 - activemq::util::URISupport, 3333
- PartialCommand
 - activemq::commands::PartialCommand, 2474
- PartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::PartialCommand, 2490
 - activemq::wireformat::openwire::marshal::v2::PartialCommand, 2478
 - activemq::wireformat::openwire::marshal::v3::PartialCommand, 2482
 - activemq::wireformat::openwire::marshal::v4::PartialCommand, 2486
 - activemq::wireformat::openwire::marshal::v5::PartialCommand, 2494
- password
 - activemq::commands::ConnectionInfo, 1216
- peek

- activemq::core::MessageDispatchChannel, 2228
- decaf::internal::util::TimerTaskHeap, 3203
- decaf::util::concurrent::SynchronousQueue, 3142
- decaf::util::PriorityQueue, 2576
- decaf::util::Queue, 2672
- peerBrokerInfos
 - activemq::commands::BrokerInfo, 782
- PERSISTENT
 - cms::DeliveryMode, 1478
- persistent
 - activemq::commands::Message, 2161
- physicalName
 - activemq::commands::ActiveMQDestination, 284
- PI
 - decaf::lang::Math, 2140
- Pointer
 - decaf::lang::Pointer, 2499, 2500
- PointerType
 - decaf::lang::Pointer, 2499
- poll
 - decaf::util::concurrent::SynchronousQueue, 3142
 - decaf::util::PriorityQueue, 2576
 - decaf::util::Queue, 2673
- PooledSession
 - activemq::cmsutil::PooledSession, 2507
- PooledThread
 - decaf::util::concurrent::PooledThread, 2518
- pop
 - decaf::util::StlQueue, 3047
- PortUnreachableException
 - decaf::net::PortUnreachableException, 2522, 2523
- position
 - decaf::nio::Buffer, 807
- POSITIVE_INFINITY
 - decaf::lang::Double, 1547
 - decaf::lang::Float, 1657
- pow
 - decaf::lang::Math, 2134
- prefetch
 - activemq::commands::ConsumerControl, 1254
- prefetchSize
 - activemq::commands::ConsumerInfo, 1307
- previous
 - decaf::util::ListIterator, 1991
- previousIndex
 - decaf::util::ListIterator, 1992
- PrimitiveList
 - activemq::util::PrimitiveList, 2527
- PrimitiveMap
 - activemq::util::PrimitiveMap, 2538
- PrimitiveType
 - activemq::util::PrimitiveValueNode, 2558
- PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2547
- PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2553
- PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2559–2561
- printStackTrace
 - cms::CMSException, 1033
 - decaf::lang::Exception, 1579
 - decaf::lang::Throwable, 3184
- priority
 - activemq::commands::ConsumerInfo, 1307
 - activemq::commands::Message, 2161
- PriorityQueue
 - decaf::util::PriorityQueue, 2573, 2574
- PriorityQueueIterator
 - decaf::util::PriorityQueue, 2578
- processBeginTransaction
 - activemq::state::CommandVisitor, 1071
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1246
- processBrokerError
 - activemq::state::CommandVisitor, 1071
 - activemq::state::CommandVisitorAdapter, 1080
- processBrokerInfo
 - activemq::state::CommandVisitor, 1071
 - activemq::state::CommandVisitorAdapter, 1080
- processCommitTransactionOnePhase
 - activemq::state::CommandVisitor, 1071
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1246
- processCommitTransactionTwoPhase
 - activemq::state::CommandVisitor, 1071
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1246
- processConnectionControl
 - activemq::state::CommandVisitor, 1071
 - activemq::state::CommandVisitorAdapter, 1080

- processConnectionError
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
- processConnectionInfo
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1246
- processConsumerControl
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
- processConsumerInfo
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processControlCommand
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
- processDestinationInfo
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processEndTransaction
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processFlushCommand
 - activemq::state::CommandVisitor, 1072
 - activemq::state::CommandVisitorAdapter, 1080
- processForgetTransaction
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
- processKeepAliveInfo
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
- processMessage
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processMessageAck
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processMessageDispatch
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
- processMessageDispatchNotification
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
- processMessagePull
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
- processPrepareTransaction
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processProducerAck
 - activemq::state::CommandVisitor, 1073
 - activemq::state::CommandVisitorAdapter, 1080
- processProducerInfo
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1247
- processRecoverTransactions
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1080
- processRemoveConnection
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1248
- processRemoveConsumer
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1080
 - activemq::state::ConnectionStateTracker, 1248
- processRemoveDestination
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1080

- activemq::state::ConnectionStateTracker, 1248
- processRemoveInfo
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1080
- processRemoveProducer
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1081
 - activemq::state::ConnectionStateTracker, 1248
- processRemoveSession
 - activemq::state::CommandVisitor, 1074
 - activemq::state::CommandVisitorAdapter, 1081
 - activemq::state::ConnectionStateTracker, 1248
- processRemoveSubscriptionInfo
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
- processReplayCommand
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
- processResponse
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
- processRollbackTransaction
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
 - activemq::state::ConnectionStateTracker, 1248
- processSessionInfo
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
 - activemq::state::ConnectionStateTracker, 1248
- processShutdownInfo
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
- processTransactionInfo
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1081
- processWireFormat
 - activemq::state::CommandVisitor, 1075
 - activemq::state::CommandVisitorAdapter, 1082
- PRODUCER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 284
- ProducerAck
 - activemq::commands::ProducerAck, 2580
- ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2600
 - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2584
 - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2588
 - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2592
 - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2596
- ProducerExecutor
 - activemq::cmsutil::CmsTemplate, 1053
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2604
- ProducerId
 - activemq::commands::ProducerId, 2607
- producerId
 - activemq::commands::Message, 2161
 - activemq::commands::MessageId, 2283
 - activemq::commands::ProducerAck, 2582
 - activemq::commands::ProducerInfo, 2635
- ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2624
 - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2612
 - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2616
 - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2628
 - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2620
- ProducerInfo
 - activemq::commands::ProducerInfo, 2632
- ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2649
 - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2637
 - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2653
 - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2645
 - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2641
- producerSequenceId
 - activemq::commands::MessageId, 2283
- ProducerState

- activemq::state::ProducerState, 2656
- Properties
 - decaf::util::Properties, 2659
- propertyExists
 - activemq::commands::ActiveMQMessageTemplate, 376
 - cms::Message, 2178
- ProtocolException
 - decaf::net::ProtocolException, 2667, 2668
- publish
 - decaf::util::logging::Handler, 1711
 - decaf::util::logging::StreamHandler, 3079
- purge
 - decaf::util::Timer, 3190
- push
 - decaf::util::StlQueue, 3047
- put
 - decaf::internal::nio::ByteBuffer, 885
 - decaf::internal::nio::CharArrayBuffer, 995, 996
 - decaf::internal::nio::DoubleArrayBuffer, 1553, 1554
 - decaf::internal::nio::FloatArrayBuffer, 1663
 - decaf::internal::nio::IntArrayBuffer, 1749
 - decaf::internal::nio::LongArrayBuffer, 2076, 2077
 - decaf::internal::nio::ShortArrayBuffer, 2920, 2921
 - decaf::internal::util::ByteArrayAdapter, 863
 - decaf::nio::ByteBuffer, 929, 930
 - decaf::nio::CharBuffer, 1008–1010
 - decaf::nio::DoubleBuffer, 1563–1565
 - decaf::nio::FloatBuffer, 1672, 1673
 - decaf::nio::IntBuffer, 1758, 1759
 - decaf::nio::LongBuffer, 2086–2088
 - decaf::nio::ShortBuffer, 2930, 2931
 - decaf::util::concurrent::ConcurrentStlMap, 1110
 - decaf::util::concurrent::SynchronousQueue, 3142
 - decaf::util::Map, 2101
 - decaf::util::StlMap, 3038
- putAll
 - decaf::util::concurrent::ConcurrentStlMap, 1111
 - decaf::util::Map, 2102
 - decaf::util::StlMap, 3038, 3039
- putChar
 - decaf::internal::nio::ByteBuffer, 885, 886
 - decaf::internal::util::ByteArrayAdapter, 864
 - decaf::nio::ByteBuffer, 931
- putDouble
 - decaf::internal::nio::ByteBuffer, 886, 887
 - decaf::internal::util::ByteArrayAdapter, 864
 - decaf::nio::ByteBuffer, 932
- putDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 865
- putFloat
 - decaf::internal::nio::ByteBuffer, 887, 888
 - decaf::internal::util::ByteArrayAdapter, 865
 - decaf::nio::ByteBuffer, 932, 933
- putFloatAt
 - decaf::internal::util::ByteArrayAdapter, 865
- putIfAbsent
 - decaf::util::concurrent::ConcurrentMap, 1098
 - decaf::util::concurrent::ConcurrentStlMap, 1111
- putInt
 - decaf::internal::nio::ByteBuffer, 888
 - decaf::internal::util::ByteArrayAdapter, 866
 - decaf::nio::ByteBuffer, 933, 934
- putIntAt
 - decaf::internal::util::ByteArrayAdapter, 866
- putLong
 - decaf::internal::nio::ByteBuffer, 889
 - decaf::internal::util::ByteArrayAdapter, 867
 - decaf::nio::ByteBuffer, 934
- putLongAt
 - decaf::internal::util::ByteArrayAdapter, 867
- putShort
 - decaf::internal::nio::ByteBuffer, 890
 - decaf::internal::util::ByteArrayAdapter, 867
 - decaf::nio::ByteBuffer, 935
- putShortAt
 - decaf::internal::util::ByteArrayAdapter, 868
- QUEUE
 - cms::Destination, 1480
- QUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- QUEUE_QUALIFIED_PREFIX

- activemq::commands::ActiveMQDestination, 284
- queueTask
 - decaf::util::concurrent::ThreadPool, 3179
- quoteIllegal
 - decaf::internal::net::URLEncoderDecoder, 3320
- Random
 - decaf::util::Random, 2678
- random
 - decaf::lang::Math, 2134
- randomUUID
 - decaf::util::UUID, 3360
- read
 - activemq::io::LoggingInputStream, 2038, 2039
 - decaf::internal::io::StandardInputStream, 3003
 - decaf::internal::util::ByteArrayAdapter, 868
 - decaf::io::BlockingByteArrayInputStream, 728
 - decaf::io::BufferedInputStream, 811, 812
 - decaf::io::ByteArrayInputStream, 896
 - decaf::io::DataInputStream, 1381, 1382
 - decaf::io::FilterInputStream, 1635, 1636
 - decaf::io::InputStream, 1741, 1742
 - decaf::io::Reader, 2683
 - decaf::net::SocketInputStream, 2980, 2981
 - decaf::nio::CharBuffer, 1011
- readAsciiString
 - activemq::wireformat::openwire::marshal::BaseDataStreamMessage, 703
- readBoolean
 - activemq::commands::ActiveMQBytesMessage, 202
 - activemq::commands::ActiveMQStreamMessage, 468
 - activemq::wireformat::openwire::utils::BooleanStream, 740
 - cms::BytesMessage, 942
 - cms::StreamMessage, 3084
 - decaf::io::DataInputStream, 1382
- readByte
 - activemq::commands::ActiveMQBytesMessage, 202
 - activemq::commands::ActiveMQStreamMessage, 469
 - cms::BytesMessage, 942
 - cms::StreamMessage, 3084
 - decaf::io::DataInputStream, 1383
 - decaf::io::Reader, 2683
- readBytes
 - activemq::commands::ActiveMQBytesMessage, 203
 - activemq::commands::ActiveMQStreamMessage, 469, 470
 - cms::BytesMessage, 943
 - cms::StreamMessage, 3084, 3085
- readChar
 - activemq::commands::ActiveMQBytesMessage, 204
 - activemq::commands::ActiveMQStreamMessage, 470
 - cms::BytesMessage, 944
 - cms::StreamMessage, 3086
 - decaf::io::DataInputStream, 1383
- ReadChecker
 - activemq::transport::inactivity::InactivityMonitor, 1736
 - activemq::transport::inactivity::ReadChecker, 2682
- readDouble
 - activemq::commands::ActiveMQBytesMessage, 204
 - activemq::commands::ActiveMQStreamMessage, 471
 - cms::BytesMessage, 944
 - cms::StreamMessage, 3086
 - decaf::io::DataInputStream, 1383
- readFloat
 - activemq::commands::ActiveMQBytesMessage, 205
 - activemq::commands::ActiveMQStreamMessage, 471
 - cms::BytesMessage, 944
 - cms::StreamMessage, 3086
 - decaf::io::DataInputStream, 1383
- readFully
 - decaf::io::DataInputStream, 1384
- readInt
 - activemq::commands::ActiveMQBytesMessage, 205
 - activemq::commands::ActiveMQStreamMessage, 471
 - cms::BytesMessage, 945
 - cms::StreamMessage, 3087
 - decaf::io::DataInputStream, 1385
- readLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2689
- readLong
 - activemq::commands::ActiveMQBytesMessage, 205
 - activemq::commands::ActiveMQStreamMessage, 472
 - cms::BytesMessage, 945

- cms::StreamMessage, 3087
- decaf::io::DataInputStream, 1385
- readOnly
 - decaf::internal::nio::CharArrayBuffer, 997
- ReadOnlyBufferException
 - decaf::nio::ReadOnlyBufferException, 2685, 2686
- readShort
 - activemq::commands::ActiveMQBytesMessage, 206
 - activemq::commands::ActiveMQStreamMessage, 472
 - cms::BytesMessage, 945
 - cms::StreamMessage, 3088
 - decaf::io::DataInputStream, 1385
- readString
 - activemq::commands::ActiveMQBytesMessage, 206
 - activemq::commands::ActiveMQStreamMessage, 473
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2468
 - cms::BytesMessage, 946
 - cms::StreamMessage, 3088
 - decaf::io::DataInputStream, 1386
- readUnsignedByte
 - decaf::io::DataInputStream, 1386
- readUnsignedShort
 - activemq::commands::ActiveMQBytesMessage, 206
 - activemq::commands::ActiveMQStreamMessage, 473
 - cms::BytesMessage, 946
 - cms::StreamMessage, 3088
 - decaf::io::DataInputStream, 1386
- readUTF
 - activemq::commands::ActiveMQBytesMessage, 207
 - cms::BytesMessage, 947
 - decaf::io::DataInputStream, 1387
- RECEIPT
 - activemq::wireformat::stomp::StompCommand, 3059
- receive
 - activemq::cmsutil::CachedConsumer, 954, 955
 - activemq::cmsutil::CmsTemplate, 1047, 1048
 - activemq::core::ActiveMQConsumer, 269
 - cms::MessageConsumer, 2215, 2216
- RECEIVE_TIMEOUT_INDEFINITE_WAIT
 - activemq::cmsutil::CmsTemplate, 1053
- RECEIVE_TIMEOUT_NO_WAIT
 - activemq::cmsutil::CmsTemplate, 1053
- ReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 1053
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2690
- receiveNoWait
 - activemq::cmsutil::CachedConsumer, 955
 - activemq::core::ActiveMQConsumer, 269
 - cms::MessageConsumer, 2216
- receiveSelected
 - activemq::cmsutil::CmsTemplate, 1048, 1049
- recievedByDFBridge
 - activemq::commands::Message, 2161
- reconnect
 - activemq::transport::failover::FailoverTransport, 1619
 - activemq::transport::IOTransport, 1826
 - activemq::transport::mock::MockTransport, 2377
 - activemq::transport::Transport, 3276
 - activemq::transport::TransportFilter, 3286
- recover
 - activemq::cmsutil::PooledSession, 2516
 - activemq::core::ActiveMQSession, 456
 - cms::Session, 2850
- redeliveryCounter
 - activemq::commands::Message, 2161
 - activemq::commands::MessageDispatch, 2223
- redispatch
 - activemq::core::ActiveMQSession, 457
- ReentrantLock
 - decaf::util::concurrent::locks::ReentrantLock, 2693
- ReferenceType
 - decaf::lang::Pointer, 2499
- registerFactory
 - activemq::transport::TransportRegistry, 3292
 - activemq::wireformat::WireFormatRegistry, 3401
- RegistrationSecurityProvider
 - decaf::security_provider::SecurityProviderMap, 2824
- RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2699, 2700
- relativize
 - decaf::net::URI, 3316
- release
 - decaf::lang::AtomicRefCounter, 639
 - decaf::lang::Pointer, 2502
 - decaf::util::concurrent::Semaphore, 2832
- releaseAll

- activemq::cmsutil::ResourceLifecycleManager, 2780
- remaining
 - decaf::nio::Buffer, 807
- remainingCapacity
 - decaf::util::concurrent::SynchronousQueue, 3143
- remove
 - activemq::util::ActiveMQProperties, 417
 - cms::CMSProperties, 1038
 - decaf::internal::util::TimerTaskHeap, 3204
 - decaf::util::AbstractCollection, 155
 - decaf::util::AbstractQueue, 164
 - decaf::util::Collection, 1060
 - decaf::util::concurrent::ConcurrentMap, 1098
 - decaf::util::concurrent::ConcurrentStlMap, 1112, 1113
 - decaf::util::concurrent::SynchronousQueue, 3143
 - decaf::util::Iterator, 1833
 - decaf::util::List, 1988
 - decaf::util::Map, 2103
 - decaf::util::PriorityQueue, 2576, 2577
 - decaf::util::Properties, 2663
 - decaf::util::Queue, 2673
 - decaf::util::StlList, 3027
 - decaf::util::StlMap, 3039
 - decaf::util::StlSet, 3055
- removeAll
 - activemq::core::MessageDispatchChannel, 2228
 - decaf::util::AbstractCollection, 155
 - decaf::util::AbstractSet, 167
 - decaf::util::Collection, 1060
 - decaf::util::concurrent::SynchronousQueue, 3143
- removeConsumer
 - activemq::state::SessionState, 2904
- removeDispatcher
 - activemq::core::ActiveMQConnection, 240
- removeHandler
 - decaf::util::logging::Logger, 2034
- RemoveInfo
 - activemq::commands::RemoveInfo, 2704
- RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 2708
 - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2712
 - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2716
 - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 2720
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2724
- removeProducer
 - activemq::core::ActiveMQConnection, 240
 - activemq::state::SessionState, 2904
- removeProperty
 - activemq::wireformat::stomp::StompFrame, 3064
- removePropertyChangeListener
 - decaf::util::logging::LogManager, 2049
- removeSession
 - activemq::core::ActiveMQConnection, 241
 - activemq::state::ConnectionState, 1242
- RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 2728
- RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 2741
 - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 2737
 - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 2745
 - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 2749
 - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2733
- removeSynchronization
 - activemq::core::ActiveMQTransactionContext, 624
- removeTask
 - activemq::threads::CompositeTaskRunner, 1093
- removeTempDestination
 - activemq::state::ConnectionState, 1242
- RemoveTransactionAction
 - activemq::state::ConnectionStateTracker, 1249
- removeTransactionState
 - activemq::state::ConnectionState, 1242
- removeTransportListener
 - activemq::core::ActiveMQConnection, 241
- removeURI
 - activemq::transport::CompositeTransport, 1095
 - activemq::transport::failover::FailoverTransport, 1619
 - activemq::transport::failover::URIPool, 3330
- RemoveInfoMarshaller, 2708
- RemoveInfoMarshaller, 2712
- RemoveInfoMarshaller, 2716
- RemoveInfoMarshaller, 2720
- replace

- decaf::util::concurrent::ConcurrentMap, 1099, 1100
- decaf::util::concurrent::ConcurrentStlMap, 1113, 1114
- ReplayCommand
 - activemq::commands::ReplayCommand, 2753
- ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 2773
 - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2757
 - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2761
 - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2769
 - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2765
- replyTo
 - activemq::commands::Message, 2161
- request
 - activemq::transport::correlator::ResponseCorrelator, 2789
 - activemq::transport::failover::FailoverTransport, 1619, 1620
 - activemq::transport::IOTransport, 1827
 - activemq::transport::logging::LoggingTransport, 2043
 - activemq::transport::mock::MockTransport, 2377
 - activemq::transport::Transport, 3276, 3277
 - activemq::transport::TransportFilter, 3286
 - activemq::wireformat::openwire::OpenWireFormat::v1::Request, 2464
- reserve
 - decaf::util::concurrent::ThreadPool, 3179
- reset
 - activemq::commands::ActiveMQBytesMessage, 207
 - activemq::commands::ActiveMQStreamMessage, 473
 - activemq::state::ConnectionState, 1243
 - cms::BytesMessage, 947
 - decaf::internal::io::StandardInputStream, 3004
 - decaf::internal::util::TimerTaskHeap, 3204
 - decaf::io::BlockingByteArrayInputStream, 728
 - decaf::io::BufferedInputStream, 812
 - decaf::io::ByteArrayInputStream, 896
 - decaf::io::ByteArrayOutputStream, 903
 - decaf::io::FilterInputStream, 1636
 - decaf::io::InputStream, 1742
 - decaf::lang::Pointer, 2502
- decaf::net::SocketInputStream, 2981
- decaf::nio::Buffer, 808
- decaf::util::StringTokenizer, 3096
- resize
 - decaf::internal::util::ByteArrayAdapter, 869
- resolve
 - decaf::net::URI, 3317
 - ReplayDestinationMarshaller, 1039
 - activemq::cmsutil::CmsDestinationAccessor, 1039
 - activemq::cmsutil::DestinationResolver, 1039
 - activemq::cmsutil::DynamicDestinationResolver, 1039
 - ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate, 1053
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2776
 - ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 1053
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2777
- ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 2778
- Response
 - activemq::commands::Response, 2782
 - ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 2788
 - ResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 2807
 - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 2792
 - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 2797
 - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 2812
 - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2802
- restore
 - activemq::state::ConnectionStateTracker, 1249
- restoreTransport
 - activemq::transport::failover::FailoverTransport, 1620
- result
 - activemq::commands::IntegerResponse, 1779
- resume
 - activemq::commands::ConnectionControl, 1139

- retainAll
 - decaf::util::AbstractCollection, 156
 - decaf::util::Collection, 1061
 - decaf::util::concurrent::SynchronousQueue, 3143
- retroactive
 - activemq::commands::ConsumerInfo, 1307
- returnInstance
 - decaf::util::logging::LogManager, 2049
 - decaf::util::logging::LogWriter, 2056
- returnRef
 - decaf::internal::nio::ByteArrayPerspective, 911
- returnSession
 - activemq::cmsutil::SessionPool, 2902
- reverse
 - decaf::lang::Integer, 1771
 - decaf::lang::Long, 2066
 - decaf::util::StlQueue, 3048
- reverseBytes
 - decaf::lang::Integer, 1771
 - decaf::lang::Long, 2066
 - decaf::lang::Short, 2912
- rewind
 - decaf::nio::Buffer, 808
- rollback
 - activemq::cmsutil::PooledSession, 2516
 - activemq::core::ActiveMQConsumer, 270
 - activemq::core::ActiveMQSession, 457
 - activemq::core::ActiveMQTransactionContext, 625
 - cms::Session, 2850
- rotateLeft
 - decaf::lang::Integer, 1771
 - decaf::lang::Long, 2066
- rotateRight
 - decaf::lang::Integer, 1772
 - decaf::lang::Long, 2066
- round
 - decaf::lang::Math, 2135
- run
 - activemq::threads::CompositeTaskRunner, 1093
 - activemq::threads::DedicatedTaskRunner, 1473
 - activemq::transport::inactivity::ReadChecker, 2682
 - activemq::transport::inactivity::WriteChecker, 3403
 - activemq::transport::IOTransport, 1827
 - activemq::transport::mock::InternalCommand, 1800
 - decaf::lang::Runnable, 2816
 - decaf::util::concurrent::PooledThread, 2519
- RuntimeException
 - decaf::lang::exceptions::RuntimeException, 2819, 2820
- schedule
 - decaf::util::Timer, 3190–3194
- scheduleAtFixedRate
 - decaf::util::Timer, 3195–3197
- scheduledExecutionTime
 - decaf::util::TimerTask, 3200
- SECONDS
 - decaf::util::concurrent::TimeUnit, 3213
- SecurityProviderRegistrar
 - decaf::security_-
provider::SecurityProviderRegistrar, 2825
- selector
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2691
 - activemq::commands::ConsumerInfo, 1307
 - activemq::commands::SubscriptionInfo, 3101
- Semaphore
 - decaf::util::concurrent::Semaphore, 2829
- semaphore
 - decaf::util::concurrent::ConditionHandle, 1124
- SEND
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- send
 - activemq::cmsutil::CachedProducer, 960, 961
 - activemq::cmsutil::CmsTemplate, 1049, 1050
 - activemq::core::ActiveMQProducer, 410, 411
 - activemq::core::ActiveMQSession, 457
 - cms::MessageProducer, 2335, 2336
- SendExecutor
 - activemq::cmsutil::CmsTemplate, 1053
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2836
- sendPullRequest
 - activemq::core::ActiveMQConnection, 241
- ServerSocket
 - decaf::net::ServerSocket, 2837
- serviceName
 - activemq::commands::DiscoveryEvent, 1514
- SESSION_TRANSACTED
 - cms::Session, 2842
- SessionId
 - activemq::commands::SessionId, 2854

- sessionId
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionInfo, 2880
- SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 2862
 - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 2866
 - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 2870
 - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 2874
 - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 2858
- SessionInfo
 - activemq::commands::SessionInfo, 2878
- SessionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 2882
 - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2890
 - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2898
 - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2886
 - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2894
- SessionPool
 - activemq::cmsutil::SessionPool, 2901
- SessionState
 - activemq::state::SessionState, 2904
- set
 - decaf::util::concurrent::atomic::AtomicBoolean, 631
 - decaf::util::concurrent::atomic::AtomicInteger, 637
 - decaf::util::concurrent::atomic::AtomicReference, 642
 - decaf::util::List, 1989
 - decaf::util::ListIterator, 1992
 - decaf::util::StlList, 3028
- setAbsolute
 - decaf::internal::net::URIType, 3343
- setAckHandler
 - activemq::commands::Message, 2157
- setAckMode
 - activemq::commands::SessionInfo, 2879
- setAckType
 - activemq::commands::MessageAck, 2192
- setAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1305
- setAdvisory
 - activemq::commands::ActiveMQDestination, 281
 - setAlwaysSyncSend
 - activemq::core::ActiveMQConnectionSupport, 257
 - setArrival
 - activemq::commands::Message, 2157
 - setAuthority
 - decaf::internal::net::URIType, 3343
 - setBackOffMultiplier
 - activemq::transport::failover::FailoverTransport, 1621
 - setBackup
 - activemq::transport::failover::FailoverTransport, 1621
 - setBackupPoolSize
 - activemq::transport::failover::BackupTransportPool, 649
 - activemq::transport::failover::FailoverTransport, 1621
 - setBlockSize
 - decaf::util::concurrent::ThreadPool, 3179
 - setBody
 - activemq::wireformat::stomp::StompFrame, 3064
 - setBodyBytes
 - activemq::commands::ActiveMQBytesMessage, 207
 - cms::BytesMessage, 947
 - setBool
 - activemq::util::PrimitiveList, 2531
 - activemq::util::PrimitiveMap, 2542
 - activemq::util::PrimitiveValueNode, 2565
 - setBoolean
 - activemq::commands::ActiveMQMapMessage, 317
 - cms::MapMessage, 2112
 - setBooleanProperty
 - activemq::commands::ActiveMQMessageTemplate, 376
 - activemq::wireformat::openwire::utils::MessagePropertyInterce, 2343
 - cms::Message, 2178
 - setBranchQualifier
 - activemq::commands::XATransactionId, 3413
 - setBrokerId
 - activemq::commands::BrokerInfo, 780
 - setBrokerInTime
 - activemq::commands::Message, 2158
 - setBrokerMasterConnector
 - activemq::commands::ConnectionInfo, 1215
 - setBrokerName

- activemq::commands::BrokerInfo, 780
- activemq::commands::DiscoveryEvent, 1513
- setBrokerOutTime
 - activemq::commands::Message, 2158
- setBrokerPath
 - activemq::commands::ConnectionInfo, 1215
 - activemq::commands::ConsumerInfo, 1305
 - activemq::commands::DestinationInfo, 1487
 - activemq::commands::Message, 2158
 - activemq::commands::ProducerInfo, 2634
- setBrokerSequenceId
 - activemq::commands::MessageId, 2282
- setBrokerUploadUrl
 - activemq::commands::BrokerInfo, 780
- setBrokerURL
 - activemq::commands::BrokerInfo, 780
 - activemq::core::ActiveMQConnectionFactory, 247
- setBrowser
 - activemq::commands::ConsumerInfo, 1305
- setBuffer
 - decaf::io::ByteArrayInputStream, 897
 - decaf::io::ByteArrayOutputStream, 903
- setByte
 - activemq::commands::ActiveMQMapMessage, 317
 - activemq::util::PrimitiveList, 2532
 - activemq::util::PrimitiveMap, 2543
 - activemq::util::PrimitiveValueNode, 2565
 - cms::MapMessage, 2112
- setByteArray
 - activemq::util::PrimitiveList, 2532
 - activemq::util::PrimitiveMap, 2543
 - activemq::util::PrimitiveValueNode, 2565
 - decaf::io::BlockingByteArrayInputStream, 729
 - decaf::io::ByteArrayInputStream, 897
- setByteProperty
 - activemq::commands::ActiveMQMessageTemplate, 376
 - activemq::wireformat::openwire::utils::MessagePropertyInterpreter, 2343
 - cms::Message, 2179
- setBytes
 - activemq::commands::ActiveMQMapMessage, 317
 - cms::MapMessage, 2113
- setCacheEnabled
 - activemq::commands::WireFormatInfo, 3376
 - activemq::wireformat::openwire::OpenWireFormat, 2456
- setCacheSize
 - activemq::commands::WireFormatInfo, 3376
 - activemq::wireformat::openwire::OpenWireFormat, 2456
- setCause
 - activemq::commands::BrokerError, 747
- setChar
 - activemq::commands::ActiveMQMapMessage, 318
 - activemq::util::PrimitiveList, 2532
 - activemq::util::PrimitiveMap, 2543
 - activemq::util::PrimitiveValueNode, 2566
 - cms::MapMessage, 2113
- setClientId
 - activemq::commands::ConnectionInfo, 1215
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::RemoveSubscriptionInfo, 2730
 - activemq::commands::SubscriptionInfo, 3100
 - activemq::core::ActiveMQConnectionSupport, 257
- setClientMaster
 - activemq::commands::ConnectionInfo, 1215
- setClose
 - activemq::commands::ConnectionControl, 1138
 - activemq::commands::ConsumerControl, 1253
- setClosed
 - activemq::transport::failover::BackupTransport, 645
- setCloseTimeout
 - activemq::core::ActiveMQConnectionSupport, 257
- setCluster
 - activemq::commands::Message, 2158
- setCMSCorrelationID
 - activemq::commands::ActiveMQMessageTemplate, 377
 - cms::Message, 2179
- setCMSDeliveryMode
 - activemq::commands::ActiveMQMessageTemplate, 377
 - cms::Message, 2180
- setCMSDestination
 - activemq::commands::ActiveMQMessageTemplate, 377

- cms::Message, 2180
- set CMSExpiration
 - activemq::commands::ActiveMQMessageTemplate, 1487
 - 377
- cms::Message, 2181
- set CMSMessageID
 - activemq::commands::ActiveMQMessageTemplate, 378
- cms::Message, 2181
- set CMSPriority
 - activemq::commands::ActiveMQMessageTemplate, 378
- cms::Message, 2181
- set CMSRedelivered
 - activemq::commands::ActiveMQMessageTemplate, 378
- cms::Message, 2182
- set CMSReplyTo
 - activemq::commands::ActiveMQMessageTemplate, 379
- cms::Message, 2182
- set CMSTimestamp
 - activemq::commands::ActiveMQMessageTemplate, 379
- cms::Message, 2183
- set CMSType
 - activemq::commands::ActiveMQMessageTemplate, 379
- cms::Message, 2183
- set Command
 - activemq::commands::ControlCommand, 1332
 - activemq::wireformat::stomp::StompFrame, 3064
- set CommandId
 - activemq::commands::BaseCommand, 655
 - activemq::commands::Command, 1066
 - activemq::commands::PartialCommand, 2475
- set Components
 - activemq::util::CompositeData, 1089
- set Compressed
 - activemq::commands::Message, 2158
- set Connection
 - activemq::commands::ActiveMQTempDestination, 501
- set ConnectionFactory
 - activemq::cmsutil::CmsAccessor, 1027
- set ConnectionId
 - activemq::commands::BrokerInfo, 780
 - activemq::commands::ConnectionError, 1162
 - activemq::commands::ConnectionInfo, 1215
- activemq::commands::ConsumerId, 1278
- activemq::commands::DestinationInfo, 1487
- activemq::commands::LocalTransactionId, 1996
- activemq::commands::ProducerId, 2609
- activemq::commands::RemoveSubscriptionInfo, 2730
- activemq::commands::SessionId, 2856
- activemq::commands::TransactionInfo, 3243
- set ConsumerId
 - activemq::commands::ConsumerControl, 1253
- activemq::commands::ConsumerInfo, 1305
- activemq::commands::MessageAck, 2192
- activemq::commands::MessageDispatch, 2222
- activemq::commands::MessageDispatchNotification, 2255
- activemq::commands::MessagePull, 2349
- set Content
 - activemq::commands::Message, 2158
- set CorrelationId
 - activemq::commands::Message, 2158
 - activemq::commands::MessagePull, 2349
- activemq::commands::Response, 2783
- set Data
 - activemq::commands::DataArrayResponse, 1358
 - activemq::commands::DataResponse, 1397
 - activemq::commands::PartialCommand, 2475
- setDataStructure
 - activemq::commands::Message, 2158
- setDefaultDestination
 - activemq::cmsutil::CmsTemplate, 1050
- setDefaultDestinationName
 - activemq::cmsutil::CmsTemplate, 1050
- set DeletedByBroker
 - activemq::commands::ActiveMQBlobMessage, 175
- set DeliveryMode
 - activemq::cmsutil::CachedProducer, 961
- activemq::cmsutil::CmsTemplate, 1050
- activemq::core::ActiveMQProducer, 412
- cms::MessageProducer, 2337
- set DeliveryPersistent
 - activemq::cmsutil::CmsTemplate, 1051
- set DeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2255
- set Destination
 - activemq::commands::ConsumerInfo, 1305

- activemq::commands::DestinationInfo, 1487
- activemq::commands::JournalQueueAck, 1836
- activemq::commands::JournalTopicAck, 1861
- activemq::commands::Message, 2158
- activemq::commands::MessageAck, 2192
- activemq::commands::MessageDispatch, 2222
- activemq::commands::MessageDispatchNotification, 2255
- activemq::commands::MessagePull, 2349
- activemq::commands::ProducerInfo, 2634
- activemq::commands::SubscriptionInfo, 3100
- setDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 1030
- setDisableMessageID
 - activemq::cmsutil::CachedProducer, 962
 - activemq::core::ActiveMQProducer, 412
 - cms::MessageProducer, 2337
- setDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 962
 - activemq::core::ActiveMQProducer, 412
 - cms::MessageProducer, 2337
- setDispatchAsync
 - activemq::commands::ConsumerInfo, 1305
 - activemq::commands::ProducerInfo, 2634
- setDouble
 - activemq::commands::ActiveMQMapMessage, 318
 - activemq::util::PrimitiveList, 2533
 - activemq::util::PrimitiveMap, 2543
 - activemq::util::PrimitiveValueNode, 2566
 - cms::MapMessage, 2113
- setDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 379
 - activemq::wireformat::openwire::utils::MessageProperty, 2343
 - cms::Message, 2184
- setDroppable
 - activemq::commands::Message, 2158
- setDuplexConnection
 - activemq::commands::BrokerInfo, 780
- setEnabled
 - activemq::transport::failover::BackupTransportPool, 649
- setenv
 - decaf::lang::System, 3147
- setException
 - activemq::commands::ConnectionError, 1162
 - activemq::commands::ExceptionResponse, 1584
 - setExceptionClass
 - activemq::commands::BrokerError, 748
 - setExceptionListener
 - activemq::core::ActiveMQConnection, 241
 - cms::Connection, 1134
 - setExclusive
 - activemq::commands::ActiveMQDestination, 281
 - activemq::commands::ConsumerInfo, 1305
 - setExit
 - activemq::commands::ConnectionControl, 1138
 - setExpiration
 - activemq::commands::Message, 2158
 - setExplicitQosEnabled
 - activemq::cmsutil::CmsTemplate, 1051
 - setFailOnClose
 - activemq::transport::mock::MockTransport, 2378
 - setFailOnKeepAliveSends
 - activemq::transport::mock::MockTransport, 2379
 - setFailOnReceiveMessage
 - activemq::transport::mock::MockTransport, 2379
 - setFailOnSendMessage
 - activemq::transport::mock::MockTransport, 2379
 - setFailOnStart
 - activemq::transport::mock::MockTransport, 2379
 - setFailOnStop
 - activemq::transport::mock::MockTransport, 2379
 - setFaultTolerant
 - activemq::commands::ConnectionControl, 1138
 - activemq::commands::BrokerInfo, 780
 - setFaultTolerantConfiguration
 - activemq::commands::BrokerInfo, 780
 - setFilter
 - decaf::util::logging::Handler, 1711
 - decaf::util::logging::Logger, 2035
 - decaf::util::logging::StreamHandler, 3079
 - setFirstMessageId
 - activemq::commands::MessageAck, 2192
 - setFirstNakNumber
 - activemq::commands::ReplayCommand, 2754
 - setFloat

- activemq::commands::ActiveMQMapMessage, 318
- activemq::util::PrimitiveList, 2533
- activemq::util::PrimitiveMap, 2544
- activemq::util::PrimitiveValueNode, 2566
- cms::MapMessage, 2114
- setFloatProperty
 - activemq::commands::ActiveMQMessageTemplate, 380
- activemq::wireformat::openwire::utils::MessagePropertyIntercept, 2343
- cms::Message, 2184
- setFlush
 - activemq::commands::ConsumerControl, 1253
- setFormatId
 - activemq::commands::XATransactionId, 3413
- setFormatter
 - decaf::util::logging::Handler, 1711
 - decaf::util::logging::StreamHandler, 3080
- setFragment
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3343
- setGlobalTransactionId
 - activemq::commands::XATransactionId, 3413
- setGroupID
 - activemq::commands::Message, 2158
- setGroupSequence
 - activemq::commands::Message, 2158
- setHost
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3344
- setInitialDelayTime
 - activemq::transport::inactivity::InactivityMonitor, 1735
- setInitialized
 - activemq::transport::failover::FailoverTransport, 1621
- setInitialReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1621
- setInputStream
 - activemq::transport::IOTransport, 1828
 - decaf::io::Reader, 2684
- setInt
 - activemq::commands::ActiveMQMapMessage, 319
 - activemq::util::PrimitiveList, 2533
 - activemq::util::PrimitiveMap, 2544
 - activemq::util::PrimitiveValueNode, 2566
 - cms::MapMessage, 2114
- setIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 380
 - activemq::wireformat::openwire::utils::MessagePropertyIntercept, 2344
 - cms::Message, 2185
- setKeepAlive
 - decaf::net::BufferedSocket, 821
 - decaf::net::Socket, 2968
 - decaf::net::TcpSocket, 3157
- setKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1736
- setLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2705
 - activemq::core::ActiveMQConsumer, 270
 - activemq::core::ActiveMQSession, 457
- setLastMessageId
 - activemq::commands::MessageAck, 2192
- setLastNakNumber
 - activemq::commands::ReplayCommand, 2754
- setLevel
 - decaf::util::logging::Handler, 1711
 - decaf::util::logging::Logger, 2035
 - decaf::util::logging::LogRecord, 2052
 - decaf::util::logging::StreamHandler, 3080
- setLimit
 - activemq::util::MemoryUsage, 2143
- setList
 - activemq::util::PrimitiveValueNode, 2566
- setLoggerName
 - decaf::util::logging::LogRecord, 2052
- setLong
 - activemq::commands::ActiveMQMapMessage, 319
 - activemq::util::PrimitiveList, 2534
 - activemq::util::PrimitiveMap, 2544
 - activemq::util::PrimitiveValueNode, 2567
 - cms::MapMessage, 2114
- setLongProperty
 - activemq::commands::ActiveMQMessageTemplate, 380
 - activemq::wireformat::openwire::utils::MessagePropertyIntercept, 2344
 - cms::Message, 2185
- setMagic
 - activemq::commands::WireFormatInfo, 3376
- setManageable
 - activemq::commands::ConnectionInfo, 1215
- setMap
 - activemq::util::PrimitiveValueNode, 2567
- setMark

- cms::CMSException, 1033
- decaf::lang::Exception, 1579
- decaf::lang::Throwable, 3184
- setMarshaledForm
 - activemq::commands::BaseDataStructure, 717
 - activemq::wireformat::MarshalAware, 2120
- setMarshaledProperties
 - activemq::commands::Message, 2158
 - activemq::commands::WireFormatInfo, 3376
- setMasterBroker
 - activemq::commands::BrokerInfo, 780
- setMaxCacheSize
 - activemq::state::ConnectionStateTracker, 1249
 - activemq::transport::failover::FailoverTransport, 1622
- setMaximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1305
- setMaxInactivityDuration
 - activemq::commands::WireFormatInfo, 3376
 - activemq::wireformat::openwire::OpenWireFormat, 2456
- setMaxInactivityDurationInitalDelay
 - activemq::commands::WireFormatInfo, 3376
- setMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2456
- setMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1622
- setMaxReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1622
- setMaxThreads
 - decaf::util::concurrent::ThreadPool, 3179
- setMessage
 - activemq::commands::BrokerError, 748
 - activemq::commands::JournalTrace, 1885
 - activemq::commands::MessageDispatch, 2222
 - decaf::lang::Exception, 1579
 - decaf::util::logging::LogRecord, 2052
- setMessageAck
 - activemq::commands::JournalQueueAck, 1836
- setMessageCount
 - activemq::commands::MessageAck, 2192
- setMessageId
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::Message, 2158
 - activemq::commands::MessageDispatchNotification, 2255
 - activemq::commands::MessagePull, 2349
- setMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 1051
- setMessageListener
 - activemq::cmsutil::CachedConsumer, 955
 - activemq::core::ActiveMQConsumer, 270
 - cms::MessageConsumer, 2216
- setMessageSequenceId
 - activemq::commands::JournalTopicAck, 1861
- setMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 1052
- setMimeType
 - activemq::commands::ActiveMQBlobMessage, 175
- setName
 - activemq::commands::ActiveMQBlobMessage, 175
- setNetworkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2395
- setNetworkConnection
 - activemq::commands::BrokerInfo, 780
- setNetworkConsumerPath
 - activemq::commands::ConsumerInfo, 1305
- setNetworkProperties
 - activemq::commands::BrokerInfo, 780
- setNetworkSubscription
 - activemq::commands::ConsumerInfo, 1305
- setNetworkTTL
 - activemq::commands::NetworkBridgeFilter, 2395
- setNoLocal
 - activemq::cmsutil::CmsTemplate, 1052
 - activemq::commands::ConsumerInfo, 1305
- setNoRangeAcks
 - activemq::commands::ConsumerInfo, 1305
- setNumReceivedMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2379
- setNumReceivedMessages
 - activemq::transport::mock::MockTransport, 2379
- setNumSentKeepAlives
 - activemq::transport::mock::MockTransport, 2379
- setNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::MockTransport, 2379
- setNumSentMessageBeforeFail

- activemq::transport::mock::MockTransport, 2379
- setNumSentMessages
 - activemq::transport::mock::MockTransport, 2379
- setObjectId
 - activemq::commands::RemoveInfo, 2705
- setOpaque
 - decaf::internal::net::URIType, 3344
- setOperationType
 - activemq::commands::DestinationInfo, 1487
- setOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1305
- setOrdered
 - activemq::commands::ActiveMQDestination, 282
- setOrderedTarget
 - activemq::commands::ActiveMQDestination, 282
- setOriginalDestination
 - activemq::commands::Message, 2158
- setOriginalTransactionId
 - activemq::commands::Message, 2158
- setOutgoingListener
 - activemq::transport::mock::MockTransport, 2379
- setOutputStream
 - activemq::transport::IOTransport, 1828
 - decaf::io::Writer, 3404
- setParameters
 - activemq::util::CompositeData, 1089
- setPassword
 - activemq::commands::ConnectionInfo, 1215
 - activemq::core::ActiveMQConnectionFactory, 247
 - activemq::core::ActiveMQConnectionSupport, 257
- setPath
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3344
- setPeerBrokerInfos
 - activemq::commands::BrokerInfo, 780
- setPersistent
 - activemq::commands::Message, 2158
- setPhysicalName
 - activemq::commands::ActiveMQDestination, 282
- setPooledThreadListener
 - decaf::util::concurrent::PooledThread, 2519
- setPort
 - decaf::internal::net::URIType, 3344
- setPreferedWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2456
- setPrefetch
 - activemq::commands::ConsumerControl, 1253
- setPrefetchSize
 - activemq::commands::ConsumerInfo, 1305
- setPrepared
 - activemq::state::TransactionState, 3266
- setPreparedResult
 - activemq::state::TransactionState, 3266
- setPriority
 - activemq::cmsutil::CachedProducer, 962
 - activemq::cmsutil::CmsTemplate, 1052
 - activemq::commands::ConsumerInfo, 1305
 - activemq::commands::Message, 2158
 - activemq::core::ActiveMQProducer, 412
 - cms::MessageProducer, 2338
- setProducerId
 - activemq::commands::Message, 2158
 - activemq::commands::MessageId, 2282
 - activemq::commands::ProducerAck, 2581
 - activemq::commands::ProducerInfo, 2634
- setProducerSequenceId
 - activemq::commands::MessageId, 2282
- setProducerWindowSize
 - activemq::core::ActiveMQConnectionSupport, 258
- setProperties
 - activemq::commands::WireFormatInfo, 3377
 - activemq::util::ActiveMQProperties, 417
 - decaf::util::logging::LogManager, 2049
- setProperty
 - activemq::util::ActiveMQProperties, 417
 - activemq::wireformat::stomp::StompFrame, 3065
 - cms::CMSProperties, 1038
 - decaf::util::Properties, 2663
- setPubSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 1030
 - activemq::cmsutil::CmsTemplate, 1052
- setQuery
 - decaf::internal::net::URIType, 3344
- setRandomize
 - activemq::transport::failover::FailoverTransport, 1622
 - activemq::transport::failover::URIPool, 3331
- setReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1736
- setReadOnly

- decaf::internal::nio::ByteBuffer, 891
- decaf::internal::nio::CharArrayBuffer, 996
- decaf::internal::nio::DoubleArrayBuffer, 1554
- decaf::internal::nio::FloatArrayBuffer, 1664
- decaf::internal::nio::IntArrayBuffer, 1750
- decaf::internal::nio::LongArrayBuffer, 2077
- decaf::internal::nio::ShortArrayBuffer, 2921
- setReadOnlyBody
 - activemq::commands::Message, 2158
- setReadOnlyProperties
 - activemq::commands::Message, 2159
- setReceiveBufferSize
 - decaf::net::BufferedSocket, 821
 - decaf::net::Socket, 2968
 - decaf::net::TcpSocket, 3157
- setReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 1052
- setRecievedByDFBBridge
 - activemq::commands::Message, 2159
- setReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1622
- setRedeliveryCounter
 - activemq::commands::Message, 2159
 - activemq::commands::MessageDispatch, 2222
- setRemoteBlobUrl
 - activemq::commands::ActiveMQBlobMessage, 175
- setReplyTo
 - activemq::commands::Message, 2159
- setResponse
 - activemq::transport::correlator::FutureResponse, 1705
- setResponseBuilder
 - activemq::transport::mock::InternalCommandSender, 1801
 - activemq::transport::mock::MockTransport, 2379
- setResponseRequired
 - activemq::commands::BaseCommand, 655
 - activemq::commands::Command, 1066
- setRestoreConsumers
 - activemq::state::ConnectionStateTracker, 1249
- setRestoreProducers
 - activemq::state::ConnectionStateTracker, 1249
- setRestoreSessions
 - activemq::state::ConnectionStateTracker, 1249
- setRestoreTransaction
 - activemq::state::ConnectionStateTracker, 1249
- setResult
 - activemq::commands::IntegerResponse, 1779
- setResume
 - activemq::commands::ConnectionControl, 1138
- setRetroactive
 - activemq::commands::ConsumerInfo, 1305
- setReuseAddress
 - decaf::net::BufferedSocket, 822
 - decaf::net::Socket, 2968
 - decaf::net::TcpSocket, 3158
- setScheduledTime
 - decaf::util::TimerTask, 3200
- setScheme
 - activemq::util::CompositeData, 1089
 - decaf::internal::net::URIType, 3344
- setSchemeSpecificPart
 - decaf::internal::net::URIType, 3345
- setSeed
 - decaf::util::Random, 2681
- setSelector
 - activemq::commands::ConsumerInfo, 1305
 - activemq::commands::SubscriptionInfo, 3100
- setSendBufferSize
 - decaf::net::BufferedSocket, 822
 - decaf::net::Socket, 2969
 - decaf::net::TcpSocket, 3158
- setSendTimeout
 - activemq::core::ActiveMQConnectionSupport, 258
 - activemq::core::ActiveMQProducer, 413
- setServerAuthority
 - decaf::internal::net::URIType, 3345
- setServerName
 - activemq::commands::DiscoveryEvent, 1513
- setSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 1027
- setSessionId
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionInfo, 2879
- setShort
 - activemq::commands::ActiveMQMapMessage, 319
 - activemq::util::PrimitiveList, 2534
 - activemq::util::PrimitiveMap, 2544
 - activemq::util::PrimitiveValueNode, 2567
 - cms::MapMessage, 2115
- setShortProperty

- activemq::commands::ActiveMQMessageTemplate, 381
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2344
- cms::Message, 2186
- setSize
 - activemq::commands::ProducerAck, 2581
- setSizePrefixDisabled
 - activemq::commands::WireFormatInfo, 3377
 - activemq::wireformat::openwire::OpenWireFormat, 2457
- setSlaveBroker
 - activemq::commands::BrokerInfo, 780
- setSoLinger
 - decaf::net::BufferedSocket, 822
 - decaf::net::Socket, 2969
 - decaf::net::TcpSocket, 3158
- setSoTimeout
 - decaf::net::BufferedSocket, 823
 - decaf::net::Socket, 2969
 - decaf::net::TcpSocket, 3158
- setSource
 - decaf::internal::net::URIType, 3345
- setSourceFile
 - decaf::util::logging::LogRecord, 2053
- setSourceFunction
 - decaf::util::logging::LogRecord, 2053
- setSourceLine
 - decaf::util::logging::LogRecord, 2053
- setStackTrace
 - decaf::lang::Exception, 1579
- setStackTraceElements
 - activemq::commands::BrokerError, 748
- setStackTraceEnabled
 - activemq::commands::WireFormatInfo, 3377
 - activemq::wireformat::openwire::OpenWireFormat, 2457
- setStart
 - activemq::commands::ConsumerControl, 1253
- setStop
 - activemq::commands::ConsumerControl, 1253
- setString
 - activemq::commands::ActiveMQMapMessage, 320
 - activemq::util::PrimitiveList, 2534
 - activemq::util::PrimitiveMap, 2544
 - activemq::util::PrimitiveValueNode, 2567
 - cms::MapMessage, 2115
- setStringProperty
 - activemq::commands::ActiveMQMessageTemplate, 381
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2344
 - cms::Message, 2186
 - setSubscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 2730
 - activemq::commands::SubscriptionInfo, 3100
 - setSubscribedDestination
 - activemq::commands::SubscriptionInfo, 3100
 - setSubscriptionName
 - activemq::commands::ConsumerInfo, 1305
 - setSubscriptionName
 - activemq::commands::JournalTopicAck, 1861
 - setSuspend
 - activemq::commands::ConnectionControl, 1138
 - setSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 270
 - setTargetConsumerId
 - activemq::commands::Message, 2159
 - setTcpNoDelay
 - decaf::net::TcpSocket, 3159
 - setTcpNoDelayEnabled
 - activemq::commands::WireFormatInfo, 3377
 - activemq::wireformat::openwire::OpenWireFormat, 2457
 - setText
 - activemq::commands::ActiveMQTextMessage, 576
 - cms::TextMessage, 3171
 - setTightEncodingEnabled
 - activemq::commands::WireFormatInfo, 3377
 - activemq::wireformat::openwire::OpenWireFormat, 2457
 - setTime
 - decaf::util::Date, 1470
 - setTimeout
 - activemq::commands::DestinationInfo, 1487
 - activemq::commands::MessagePull, 2349
 - activemq::transport::failover::FailoverTransport, 1622
 - setTimestamp
 - activemq::commands::Message, 2159
 - decaf::util::logging::LogRecord, 2053
 - setTimeToLive
 - activemq::cmsutil::CachedProducer, 963

- activemq::cmsutil::CmsTemplate, 1052
- activemq::core::ActiveMQProducer, 413
- cms::MessageProducer, 2338
- setTrackMessages
 - activemq::state::ConnectionStateTracker, 1249
 - activemq::transport::failover::FailoverTransport, 1622
- setTrackTransactions
 - activemq::state::ConnectionStateTracker, 1249
- setTransactionId
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::JournalTransaction, 1908
 - activemq::commands::Message, 2159
 - activemq::commands::MessageAck, 2192
 - activemq::commands::TransactionInfo, 3243
- setTransport
 - activemq::transport::failover::BackupTransport, 645
 - activemq::transport::mock::InternalCommandListerner, 1801
- setTransportListener
 - activemq::transport::failover::FailoverTransport, 1622
 - activemq::transport::IOTransport, 1828
 - activemq::transport::mock::MockTransport, 2380
 - activemq::transport::Transport, 3277
 - activemq::transport::TransportFilter, 3287
- setTreadId
 - decaf::util::logging::LogRecord, 2053
- setType
 - activemq::commands::JournalTransaction, 1908
 - activemq::commands::Message, 2159
 - activemq::commands::TransactionInfo, 3243
- setUri
 - activemq::transport::failover::BackupTransport, 645
- setUsage
 - activemq::util::MemoryUsage, 2143
- setUseAsyncSend
 - activemq::core::ActiveMQConnectionSupport, 258
- setUseExponentialBackOff
 - activemq::transport::failover::FailoverTransport, 1622
- setUseParentHandlers
 - decaf::util::logging::Logger, 2035
- setUserID
 - activemq::commands::Message, 2159
- setUserInfo
 - decaf::internal::net::URIType, 3345
- setUserName
 - activemq::commands::ConnectionInfo, 1215
- setUsername
 - activemq::core::ActiveMQConnectionFactory, 248
 - activemq::core::ActiveMQConnectionSupport, 258
- setValid
 - decaf::internal::net::URIType, 3345
- setValue
 - activemq::commands::BrokerId, 753
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::ProducerId, 2609
 - activemq::commands::SessionId, 2856
 - activemq::util::PrimitiveValueNode, 2567
 - decaf::util::Map::Entry, 1570
- setVersion
 - activemq::commands::WireFormatInfo, 3377
 - activemq::wireformat::openwire::OpenWireFormat, 2457
 - activemq::wireformat::stomp::StompWireFormat, 3073
 - activemq::wireformat::WireFormat, 3365
- setWasPrepared
 - activemq::commands::JournalTransaction, 1908
- setWindowSize
 - activemq::commands::ProducerInfo, 2634
- setWireFormat
 - activemq::transport::failover::FailoverTransport, 1622
 - activemq::transport::IOTransport, 1828
 - activemq::transport::mock::MockTransport, 2380
 - activemq::transport::Transport, 3278
 - activemq::transport::TransportFilter, 3287
- setWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1736
- Short
 - decaf::lang::Short, 2907
- SHORT_TYPE
 - activemq::util::PrimitiveValueNode, 2559
- ShortArrayBuffer

- decaf::internal::nio::ShortArrayBuffer, 2916, 2917
- ShortBuffer
 - decaf::nio::ShortBuffer, 2925
- shortValue
 - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2552
 - decaf::lang::Byte, 846
 - decaf::lang::Character, 988
 - decaf::lang::Double, 1545
 - decaf::lang::Float, 1655
 - decaf::lang::Integer, 1772
 - decaf::lang::Long, 2067
 - decaf::lang::Number, 2434
 - decaf::lang::Short, 2912
- shutdown
 - activemq::state::ConnectionState, 1243
 - activemq::state::SessionState, 2904
 - activemq::state::TransactionState, 3266
 - activemq::threads::CompositeTaskRunner, 1093
 - activemq::threads::DedicatedTaskRunner, 1473, 1474
 - activemq::threads::TaskRunner, 3150
- ShutdownInfo
 - activemq::commands::ShutdownInfo, 2935
- ShutdownInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 2938
 - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 2950
 - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 2946
 - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 2942
 - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2954
- shutdownLibrary
 - activemq::library::ActiveMQCPP, 273
- shutdownRuntime
 - decaf::lang::Runtime, 2818
- shutdownTransport
 - activemq::core::ActiveMQConnectionSupport, 258
- signal
 - decaf::util::concurrent::locks::Condition, 1123
- signalAll
 - decaf::util::concurrent::locks::Condition, 1123
- SignatureException
 - decaf::security::SignatureException, 2957, 2958
- signum
- decaf::lang::Integer, 1772
- decaf::lang::Long, 2067
- decaf::lang::Math, 2135, 2136
- SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 2960
- SimpleLogger
 - decaf::util::logging::SimpleLogger, 2962
- SIZE
 - decaf::lang::Byte, 848
 - decaf::lang::Character, 989
 - decaf::lang::Double, 1547
 - decaf::lang::Float, 1657
 - decaf::lang::Integer, 1776
 - decaf::lang::Long, 2070
 - decaf::lang::Short, 2914
- size
 - activemq::commands::ProducerAck, 2582
 - activemq::core::MessageDispatchChannel, 2228
 - activemq::wireformat::openwire::utils::HexTable, 1716
 - decaf::internal::util::TimerTaskHeap, 3204
 - decaf::io::ByteArrayOutputStream, 903
 - decaf::io::DataOutputStream, 1390
 - decaf::util::Collection, 1061
 - decaf::util::concurrent::ConcurrentStlMap, 1114
 - decaf::util::concurrent::ConcurrentStlQueue, 1114
 - decaf::util::PriorityQueue, 2577
 - decaf::util::StlList, 3028
 - decaf::util::StlMap, 3039
 - decaf::util::StlQueue, 3048
 - decaf::util::StlSet, 3056
- skip
 - decaf::internal::io::StandardInputStream, 3004
 - decaf::io::BlockingByteArrayInputStream, 729
 - decaf::io::BufferedInputStream, 812
 - decaf::io::ByteArrayInputStream, 897
 - decaf::io::DataInputStream, 1387
 - decaf::io::FilterInputStream, 1637
 - decaf::io::InputStream, 1743
 - decaf::net::SocketInputStream, 2981
- slaveBroker
 - activemq::commands::BrokerInfo, 782
- sleep
 - decaf::util::concurrent::TimeUnit, 3209
- slice
 - decaf::internal::nio::ByteBuffer, 891
 - decaf::internal::nio::CharArrayBuffer, 996

- decaf::internal::nio::DoubleArrayBuffer, 1554
- decaf::internal::nio::FloatArrayBuffer, 1664
- decaf::internal::nio::IntArrayBuffer, 1750
- decaf::internal::nio::LongArrayBuffer, 2077
- decaf::internal::nio::ShortArrayBuffer, 2921
- decaf::nio::ByteBuffer, 936
- decaf::nio::CharBuffer, 1011
- decaf::nio::DoubleBuffer, 1565
- decaf::nio::FloatBuffer, 1674
- decaf::nio::IntBuffer, 1760
- decaf::nio::LongBuffer, 2088
- decaf::nio::ShortBuffer, 2932
- SocketAddress
 - decaf::net::ServerSocket, 2837
 - decaf::net::Socket, 2965
- SocketException
 - decaf::net::SocketException, 2972, 2973
- SocketHandle
 - decaf::net::ServerSocket, 2837
 - decaf::net::Socket, 2965
- SocketInputStream
 - decaf::net::SocketInputStream, 2978
- SocketOutputStream
 - decaf::net::SocketOutputStream, 2985
- SocketTimeoutException
 - decaf::net::SocketTimeoutException, 2990, 2991
- sqrt
 - decaf::lang::Math, 2137
- src/main/activemq/cmsutil/CachedConsumer.h, 3435
- src/main/activemq/cmsutil/CachedProducer.h, 3436
- src/main/activemq/cmsutil/CmsAccessor.h, 3437
- src/main/activemq/cmsutil/CmsDestinationAccessor.h, 3438
- src/main/activemq/cmsutil/CmsTemplate.h, 3439
- src/main/activemq/cmsutil/DestinationResolver.h, 3440
- src/main/activemq/cmsutil/DynamicDestinationResolver.h, 3441
- src/main/activemq/cmsutil/MessageCreator.h, 3442
- src/main/activemq/cmsutil/PooledSession.h, 3443
- src/main/activemq/cmsutil/ProducerCallback.h, 3444
- src/main/activemq/cmsutil/ResourceLifecycleManager.h, 3445
- src/main/activemq/cmsutil/SessionCallback.h, 3446
- src/main/activemq/cmsutil/SessionPool.h, 3447
- src/main/activemq/commands/ActiveMQBlobMessage.h, 3448
- src/main/activemq/commands/ActiveMQBytesMessage.h, 3449
- src/main/activemq/commands/ActiveMQDestination.h, 3450
- src/main/activemq/commands/ActiveMQMapMessage.h, 3451
- src/main/activemq/commands/ActiveMQMessage.h, 3452
- src/main/activemq/commands/ActiveMQMessageTemplate.h, 3453
- src/main/activemq/commands/ActiveMQObjectMessage.h, 3454
- src/main/activemq/commands/ActiveMQQueue.h, 3455
- src/main/activemq/commands/ActiveMQStreamMessage.h, 3456
- src/main/activemq/commands/ActiveMQTempDestination.h, 3457
- src/main/activemq/commands/ActiveMQTempQueue.h, 3458
- src/main/activemq/commands/ActiveMQTempTopic.h, 3459
- src/main/activemq/commands/ActiveMQTextMessage.h, 3460
- src/main/activemq/commands/ActiveMQTopic.h, 3461
- src/main/activemq/commands/BaseCommand.h, 3462
- src/main/activemq/commands/BaseDataStructure.h, 3463
- src/main/activemq/commands/BooleanExpression.h, 3464
- src/main/activemq/commands/BrokerError.h, 3465
- src/main/activemq/commands/BrokerId.h, 3466
- src/main/activemq/commands/BrokerInfo.h, 3467
- src/main/activemq/commands/Command.h, 3468
- src/main/activemq/commands/ConnectionControl.h, 3469
- src/main/activemq/commands/ConnectionError.h, 3470
- src/main/activemq/commands/ConnectionId.h, 3471
- src/main/activemq/commands/ConnectionInfo.h, 3472

src/main/activemq/commands/ConsumerControl.h,	src/main/activemq/commands/ProducerAck.h,
3473	3501
src/main/activemq/commands/ConsumerId.h,	src/main/activemq/commands/ProducerId.h,
3474	3502
src/main/activemq/commands/ConsumerInfo.h,	src/main/activemq/commands/ProducerInfo.h,
3475	3503
src/main/activemq/commands/ControlCommands.h,	src/main/activemq/commands/RemoveInfo.h,
3476	3504
src/main/activemq/commands/DataArrayResponse.h,	src/main/activemq/commands/RemoveSubscriptionInfo.h,
3477	3505
src/main/activemq/commands/DataResponse.h,	src/main/activemq/commands/ReplayCommand.h,
3478	3506
src/main/activemq/commands/DataStructure.h,	src/main/activemq/commands/Response.h,
3479	3507
src/main/activemq/commands/DestinationInfo.h,	src/main/activemq/commands/SessionId.h,
3480	3508
src/main/activemq/commands/DiscoveryEvent.h,	src/main/activemq/commands/SessionInfo.h,
3481	3509
src/main/activemq/commands/ExceptionResponse.h,	src/main/activemq/commands/ShutdownInfo.h,
3482	3510
src/main/activemq/commands/FlushCommand.h,	src/main/activemq/commands/SubscriptionInfo.h,
3483	3511
src/main/activemq/commands/IntegerResponse.h,	src/main/activemq/commands/TransactionId.h,
3484	3512
src/main/activemq/commands/JournalQueueAck.h,	src/main/activemq/commands/TransactionInfo.h,
3485	3513
src/main/activemq/commands/JournalTopicAck.h,	src/main/activemq/commands/WireFormatInfo.h,
3486	3514
src/main/activemq/commands/JournalTrace.h,	src/main/activemq/commands/XATransactionId.h,
3487	3515
src/main/activemq/commands/JournalTransactions.h,	src/main/activemq/core/ActiveMQAckHandler.h,
3488	3516
src/main/activemq/commands/KeepAliveInfo.h,	src/main/activemq/core/ActiveMQConnection.h,
3489	3517
src/main/activemq/commands/LastPartialCommand.h,	src/main/activemq/core/ActiveMQConnectionFactory.h,
3490	3518
src/main/activemq/commands/LocalTransactionId.h,	src/main/activemq/core/ActiveMQConnectionMetaData.h,
3491	3519
src/main/activemq/commands/Message.h,	src/main/activemq/core/ActiveMQConnectionSupport.h,
3492	3520
src/main/activemq/commands/MessageAck.h,	src/main/activemq/core/ActiveMQConstants.h,
3494	3521
src/main/activemq/commands/MessageDispatch.h,	src/main/activemq/core/ActiveMQConsumer.h,
3495	3522
src/main/activemq/commands/MessageDispatchNotFinal.h,	src/main/activemq/core/ActiveMQProducer.h,
3496	3523
src/main/activemq/commands/MessageId.h,	src/main/activemq/core/ActiveMQSession.h,
3497	3524
src/main/activemq/commands/MessagePull.h,	src/main/activemq/core/ActiveMQSessionExecutor.h,
3498	3525
src/main/activemq/commands/NetworkBridgeFilter.h,	src/main/activemq/core/ActiveMQTransactionContext.h,
3499	3526
src/main/activemq/commands/PartialCommand.h,	src/main/activemq/core/DispatchData.h,
3500	3527
	src/main/activemq/core/Dispatcher.h,
	3528

- src/main/activemq/core/MessageDispatchChannel.h, 3563
- 3529
- src/main/activemq/core/Synchronization.h, 3564
- 3530
- src/main/activemq/exceptions/ActiveMQException.h, 3565
- 3531
- src/main/activemq/exceptions/BrokerException.h, 3566
- 3532
- src/main/activemq/exceptions/ExceptionDefines.h, 3567
- 3533
- src/main/activemq/io/LoggingInputStream.h, 3568
- 3539
- src/main/activemq/io/LoggingOutputStream.h, 3569
- 3540
- src/main/activemq/library/ActiveMQCPP.h, 3570
- 3541
- src/main/activemq/state/CommandVisitor.h, 3571
- 3542
- src/main/activemq/state/CommandVisitorAdapter.h, 3572
- 3543
- src/main/activemq/state/ConnectionState.h, 3573
- 3545
- src/main/activemq/state/ConnectionStateTracker.h, 3574
- 3546
- src/main/activemq/state/ConsumerState.h, 3575
- 3547
- src/main/activemq/state/ProducerState.h, 3576
- 3548
- src/main/activemq/state/SessionState.h, 3549
- src/main/activemq/state/Tracked.h, 3550
- src/main/activemq/state/TransactionState.h, 3551
- src/main/activemq/threads/CompositeTask.h, 3552
- src/main/activemq/threads/CompositeTaskRunner.h, 3553
- src/main/activemq/threads/DedicatedTaskRunner.h, 3554
- src/main/activemq/threads/Task.h, 3555
- src/main/activemq/threads/TaskRunner.h, 3556
- src/main/activemq/transport/AbstractTransportFactory.h, 3557
- src/main/activemq/transport/CompositeTransport.h, 3558
- src/main/activemq/transport/correlator/FutureResponse.h, 3559
- src/main/activemq/transport/correlator/ResponseCorrelator.h, 3560
- src/main/activemq/transport/DefaultTransportListener.h, 3561
- src/main/activemq/transport/failover/BackupTransport.h, 3562
- src/main/activemq/transport/failover/BackupTransportPool.h, 3563
- src/main/activemq/transport/failover/CloseTransportsTask.h, 3564
- src/main/activemq/transport/failover/FailoverTransport.h, 3565
- src/main/activemq/transport/failover/FailoverTransportFactory.h, 3566
- src/main/activemq/transport/failover/FailoverTransportListener.h, 3567
- src/main/activemq/transport/failover/URIPool.h, 3568
- src/main/activemq/transport/inactivity/InactivityMonitor.h, 3569
- src/main/activemq/transport/inactivity/ReadChecker.h, 3570
- src/main/activemq/transport/inactivity/WriteChecker.h, 3571
- src/main/activemq/transport/IOTransport.h, 3572
- src/main/activemq/transport/logging/LoggingTransport.h, 3573
- src/main/activemq/transport/mock/InternalCommandListener.h, 3574
- src/main/activemq/transport/mock/MockTransport.h, 3575
- src/main/activemq/transport/mock/MockTransportFactory.h, 3576
- src/main/activemq/transport/mock/ResponseBuilder.h, 3577
- src/main/activemq/transport/tcp/TcpTransport.h, 3578
- src/main/activemq/transport/tcp/TcpTransportFactory.h, 3579
- src/main/activemq/transport/Transport.h, 3580
- src/main/activemq/transport/TransportFactory.h, 3581
- src/main/activemq/transport/TransportFilter.h, 3582
- src/main/activemq/transport/TransportListener.h, 3583
- src/main/activemq/transport/TransportRegistry.h, 3584
- src/main/activemq/util/ActiveMQProperties.h, 3585
- src/main/activemq/util/CMSExceptionSupport.h, 3586
- src/main/activemq/util/CompositeData.h, 3588
- src/main/activemq/util/Config.h, 3589
- src/main/activemq/util/LongSequenceGenerator.h, 3590
- src/main/activemq/util/MemoryUsage.h, 3593
- src/main/activemq/util/PrimitiveList.h, 3594

3834	3660
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/ActiveMQT
3839	3665
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/BaseComm
3844	3670
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/BrokerIdMa
3849	3675
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/BrokerInfoM
3854	3680
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/Connection
3859	3685
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/Connection
3864	3690
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/Connection
3869	3695
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/Connection
3874	3700
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/ConsumerC
3879	3705
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/ConsumerId
3884	3710
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/ConsumerIn
3889	3715
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/ControlCom
3894	3720
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/DataArrayF
3899	3725
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/DataRespor
3904	3730
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/Destination
3909	3735
src/main/activemq/wireformat/openwire/marshaller/v1/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/DiscoveryE
3605	3740
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/ExceptionR
3610	3745
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/FlushComm
3615	3750
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/IntegerResp
3620	3755
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/JournalQue
3625	3760
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/JournalTop
3630	3765
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/JournalTra
3635	3770
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/JournalTra
3640	3775
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/KeepAliveIn
3645	3780
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/LastPartial
3650	3785
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/LocalTrans
3655	3790
src/main/activemq/wireformat/openwire/marshaller/v2/main/v2/BlockMessageMarshaller	src/main/activemq/wireformat/openwire/marshaller/v2/MarshallerF

3795	3621
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQC	
3800	3626
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQC	
3805	3631
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQC	
3810	3636
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQS	
3815	3641
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3820	3646
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3825	3651
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3830	3656
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3835	3661
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3840	3666
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3845	3671
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3850	3676
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3855	3681
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3860	3686
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3865	3691
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3870	3696
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3875	3701
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3880	3706
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3885	3711
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3890	3716
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3895	3721
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3900	3726
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3905	3731
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3910	3736
src/main/activemq/wireformat/openwire/marshals/v2/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3606	3741
src/main/activemq/wireformat/openwire/marshals/v3/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3611	3746
src/main/activemq/wireformat/openwire/marshals/v3/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	
3616	3751
src/main/activemq/wireformat/openwire/marshals/v3/InactiveMQWireFormat/openwire/marshal/v3/ActiveMQT	

3756
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v3/Transaction
 3761
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v3/Transaction
 3766
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v3/WireFormat
 3771
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v3/XATransaction
 3776
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQB
 3781
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQB
 3786
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQD
 3791
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQM
 3796
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQM
 3801
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQC
 3806
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQC
 3811
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQS
 3816
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQT
 3821
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQT
 3826
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQT
 3831
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQT
 3836
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ActiveMQT
 3841
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/BaseComm
 3846
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/BrokerIdMa
 3851
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/BrokerInfo
 3856
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/Connection
 3861
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/Connection
 3866
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/Connection
 3871
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/Connection
 3876
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ConsumerC
 3881
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ConsumerI
 3886
 src/main/activemq/wireformat/openwire/marshaller/v3/Main/ActiveMQWireMarshaller/openwire/marshaller/v4/ConsumerIn

3717	3852
src/main/activemq/wireformat/openwire/marshaller/v4/CommandQueueWireMarshaller/openwire/marshaller/v4/RemoveInfo	
3722	3857
src/main/activemq/wireformat/openwire/marshaller/v4/DataActiveResponseMarshaller/openwire/marshaller/v4/RemoveSub	
3727	3862
src/main/activemq/wireformat/openwire/marshaller/v4/DataResponseMarshaller/openwire/marshaller/v4/ReplayCom	
3732	3867
src/main/activemq/wireformat/openwire/marshaller/v4/DestinationInfoWireMarshaller/openwire/marshaller/v4/ResponseM	
3737	3872
src/main/activemq/wireformat/openwire/marshaller/v4/DisconnectWireMarshaller/openwire/marshaller/v4/SessionIdM	
3742	3877
src/main/activemq/wireformat/openwire/marshaller/v4/ExceptionResponseMarshaller/openwire/marshaller/v4/SessionInfo	
3747	3882
src/main/activemq/wireformat/openwire/marshaller/v4/FlushActiveMQWireMarshaller/openwire/marshaller/v4/ShutdownIn	
3752	3887
src/main/activemq/wireformat/openwire/marshaller/v4/IntegerResponseMarshaller/openwire/marshaller/v4/Subscription	
3757	3892
src/main/activemq/wireformat/openwire/marshaller/v4/JainBatchQueueWireMarshaller/openwire/marshaller/v4/Transaction	
3762	3897
src/main/activemq/wireformat/openwire/marshaller/v4/JainBatchTopicWireMarshaller/openwire/marshaller/v4/Transaction	
3767	3902
src/main/activemq/wireformat/openwire/marshaller/v4/JainBatchTranspWireMarshaller/openwire/marshaller/v4/WireFormat	
3772	3907
src/main/activemq/wireformat/openwire/marshaller/v4/JainBatchTranspWireMarshaller/openwire/marshaller/v4/XATransact	
3777	3912
src/main/activemq/wireformat/openwire/marshaller/v4/KeepActiveInfoMarshaller/openwire/marshaller/v5/ActiveMQB	
3782	3608
src/main/activemq/wireformat/openwire/marshaller/v4/LastBatchQueueWireMarshaller/openwire/marshaller/v5/ActiveMQB	
3787	3613
src/main/activemq/wireformat/openwire/marshaller/v4/LastBatchTranspWireMarshaller/openwire/marshaller/v5/ActiveMQD	
3792	3618
src/main/activemq/wireformat/openwire/marshaller/v4/MarshallerFactory/openwire/marshaller/v5/ActiveMQM	
3797	3623
src/main/activemq/wireformat/openwire/marshaller/v4/MisageActiveMarshaller/openwire/marshaller/v5/ActiveMQM	
3802	3628
src/main/activemq/wireformat/openwire/marshaller/v4/MisageDispatchMarshaller/openwire/marshaller/v5/ActiveMQC	
3807	3633
src/main/activemq/wireformat/openwire/marshaller/v4/MisageDispatchInfoMarshaller/openwire/marshaller/v5/ActiveMQC	
3812	3638
src/main/activemq/wireformat/openwire/marshaller/v4/MisageQueueWireMarshaller/openwire/marshaller/v5/ActiveMQS	
3817	3643
src/main/activemq/wireformat/openwire/marshaller/v4/MisageQueueWireMarshaller/openwire/marshaller/v5/ActiveMQT	
3822	3648
src/main/activemq/wireformat/openwire/marshaller/v4/MisageQueueWireMarshaller/openwire/marshaller/v5/ActiveMQT	
3827	3653
src/main/activemq/wireformat/openwire/marshaller/v4/NewBridgeFilterMarshaller/openwire/marshaller/v5/ActiveMQT	
3832	3658
src/main/activemq/wireformat/openwire/marshaller/v4/PartialQueueWireMarshaller/openwire/marshaller/v5/ActiveMQT	
3837	3663
src/main/activemq/wireformat/openwire/marshaller/v4/PartialQueueWireMarshaller/openwire/marshaller/v5/ActiveMQT	
3842	3668
src/main/activemq/wireformat/openwire/marshaller/v4/PartialQueueWireMarshaller/openwire/marshaller/v5/BaseComm	
3847	3673
src/main/activemq/wireformat/openwire/marshaller/v4/PartialQueueWireMarshaller/openwire/marshaller/v5/BrokerIdMa	

Generated on Fri Apr 30 14:04:20 2010 for activemq-cpp-3.1.2 by Doxygen

- 3920
- src/main/activemq/wireformat/openwire/Utils/OpenWireStringsSupportable.h, 3961
- 3921
- src/main/activemq/wireformat/stomp/StompCommandAndConstants.h, 3963
- 3922
- src/main/activemq/wireformat/stomp/StompFrame.h, 3965
- 3923
- src/main/activemq/wireformat/stomp/StompHelper.h, 3966
- 3924
- src/main/activemq/wireformat/stomp/StompWireFormat.h, 3968
- 3925
- src/main/activemq/wireformat/stomp/StompWireFormatFactory.h, 3970
- 3926
- src/main/activemq/wireformat/WireFormat.h, 3972
- 3927
- src/main/activemq/wireformat/WireFormatFactory.h, 3973
- 3928
- src/main/activemq/wireformat/WireFormatNegotiator.h, 3974
- 3929
- src/main/activemq/wireformat/WireFormatRegistry.h, 3975
- 3930
- src/main/cms/BytesMessage.h, 3931
- src/main/cms/Closeable.h, 3932
- src/main/cms/CMSException.h, 3934
- src/main/cms/CMSProperties.h, 3935
- src/main/cms/CMSSecurityException.h, 3936
- src/main/cms/Config.h, 3590
- src/main/cms/Connection.h, 3937
- src/main/cms/ConnectionFactory.h, 3938
- src/main/cms/ConnectionMetaData.h, 3939
- src/main/cms/DeliveryMode.h, 3940
- src/main/cms/Destination.h, 3941
- src/main/cms/ExceptionListener.h, 3942
- src/main/cms/IllegalStateException.h, 3943
- src/main/cms/InvalidClientIdException.h, 3945
- src/main/cms/InvalidDestinationException.h, 3946
- src/main/cms/InvalidSelectorException.h, 3947
- src/main/cms/MapMessage.h, 3948
- src/main/cms/Message.h, 3949
- src/main/cms/MessageConsumer.h, 3949
- src/main/cms/MessageEOFException.h, 3950
- src/main/cms/MessageFormatException.h, 3951
- src/main/cms/MessageListener.h, 3952
- src/main/cms/MessageNotReadableException.h, 3953
- src/main/cms/MessageNotWritableException.h, 3954
- src/main/cms/MessageProducer.h, 3955
- src/main/cms/ObjectMessage.h, 3956
- src/main/cms/Queue.h, 3957
- src/main/cms/QueueBrowser.h, 3959
- src/main/cms/Session.h, 3960
- src/main/cms/Stopable.h, 3961
- src/main/cms/Stopable.h, 3962
- src/main/cms/TemporaryQueue.h, 3964
- src/main/cms/TemporaryTopic.h, 3965
- src/main/cms/TextMessage.h, 3966
- src/main/cms/Topic.h, 3967
- src/main/cms/UnsupportedOperationException.h, 3968
- src/main/decaf/internal/AprPool.h, 3970
- src/main/decaf/internal/DecafRuntime.h, 3971
- src/main/decaf/internal/io/StandardErrorOutputStream.h, 3972
- src/main/decaf/internal/io/StandardInputStream.h, 3973
- src/main/decaf/internal/io/StandardOutputStream.h, 3974
- src/main/decaf/internal/net/URIEncoderDecoder.h, 3975
- src/main/decaf/internal/net/URIHelper.h, 3976
- src/main/decaf/internal/net/URIType.h, 3977
- src/main/decaf/internal/nio/BufferFactory.h, 3978
- src/main/decaf/internal/nio/ByteBuffer.h, 3979
- src/main/decaf/internal/nio/ByteBufferPerspective.h, 3980
- src/main/decaf/internal/nio/CharArrayBuffer.h, 3981
- src/main/decaf/internal/nio/DoubleArrayBuffer.h, 3982
- src/main/decaf/internal/nio/FloatArrayBuffer.h, 3983
- src/main/decaf/internal/nio/IntArrayBuffer.h, 3984
- src/main/decaf/internal/nio/LongArrayBuffer.h, 3985
- src/main/decaf/internal/nio/ShortArrayBuffer.h, 3986
- src/main/decaf/internal/util/ByteArrayAdapter.h, 3987
- src/main/decaf/internal/util/concurrent/ConditionImpl.h, 3988
- src/main/decaf/internal/util/concurrent/MutexImpl.h, 3989
- src/main/decaf/internal/util/concurrent/SynchronizableImpl.h, 3990
- src/main/decaf/internal/util/concurrent/Transferer.h, 3991
- src/main/decaf/internal/util/concurrent/TransferQueue.h, 3992

- src/main/decaf/internal/util/concurrent/TransferStack.h, 3993
- src/main/decaf/internal/util/concurrent/unix/Condition.h, 3994
- src/main/decaf/internal/util/concurrent/unix/MutexHandler.h, 3996
- src/main/decaf/internal/util/concurrent/windows/ConditionHandler.h, 3995
- src/main/decaf/internal/util/concurrent/windows/MutexHandler.h, 3997
- src/main/decaf/internal/util/HexStringParser.h, 3998
- src/main/decaf/internal/util/TimerTaskHeap.h, 3999
- src/main/decaf/io/BlockingByteArrayInputStream.h, 4000
- src/main/decaf/io/BufferedInputStream.h, 4001
- src/main/decaf/io/BufferedOutputStream.h, 4002
- src/main/decaf/io/ByteArrayInputStream.h, 4003
- src/main/decaf/io/ByteArrayOutputStream.h, 4004
- src/main/decaf/io/Closeable.h, 3933
- src/main/decaf/io/DataInputStream.h, 4005
- src/main/decaf/io/DataOutputStream.h, 4006
- src/main/decaf/io/EOFException.h, 4007
- src/main/decaf/io/FilterInputStream.h, 4008
- src/main/decaf/io/FilterOutputStream.h, 4009
- src/main/decaf/io/InputStream.h, 4010
- src/main/decaf/io/InterruptedIOException.h, 4011
- src/main/decaf/io/IOException.h, 4012
- src/main/decaf/io/OutputStream.h, 4013
- src/main/decaf/io/Reader.h, 4014
- src/main/decaf/io/UnsupportedEncodingException.h, 4015
- src/main/decaf/io/UTFDataFormatException.h, 4016
- src/main/decaf/io/Writer.h, 4017
- src/main/decaf/lang/Appendable.h, 4018
- src/main/decaf/lang/Boolean.h, 4019
- src/main/decaf/lang/Byte.h, 4020
- src/main/decaf/lang/Character.h, 4021
- src/main/decaf/lang/CharSequence.h, 4022
- src/main/decaf/lang/Comparable.h, 4023
- src/main/decaf/lang/Double.h, 4024
- src/main/decaf/lang/Exception.h, 4025
- src/main/decaf/lang/exceptions/ClassCastException.h, 4026
- src/main/decaf/lang/exceptions/ExceptionDefinitions.h, 3536
- src/main/decaf/lang/exceptions/IllegalArgumentException.h, 4027
- src/main/decaf/lang/exceptions/IllegalMonitorStateException.h, 4028
- src/main/decaf/lang/exceptions/IllegalStateException.h, 3944
- src/main/decaf/lang/exceptions/IllegalThreadStateException.h, 4029
- src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h, 4030
- src/main/decaf/lang/exceptions/InterruptedException.h, 4031
- src/main/decaf/lang/exceptions/InvalidStateException.h, 4032
- src/main/decaf/lang/exceptions/NoSuchElementException.h, 4033
- src/main/decaf/lang/exceptions/NullPointerException.h, 4034
- src/main/decaf/lang/exceptions/NumberFormatException.h, 4035
- src/main/decaf/lang/exceptions/RuntimeException.h, 4036
- src/main/decaf/lang/exceptions/UnsupportedOperationException.h, 3969
- src/main/decaf/lang/Float.h, 4037
- src/main/decaf/lang/Integer.h, 4038
- src/main/decaf/lang/Iterable.h, 4039
- src/main/decaf/lang/Long.h, 4040
- src/main/decaf/lang/Math.h, 4041
- src/main/decaf/lang/Number.h, 4042
- src/main/decaf/lang/Pointer.h, 4043
- src/main/decaf/lang/Runnable.h, 4045
- src/main/decaf/lang/Runtime.h, 4046
- src/main/decaf/lang/Short.h, 4047
- src/main/decaf/lang/System.h, 4048
- src/main/decaf/lang/Thread.h, 4049
- src/main/decaf/lang/ThreadGroup.h, 4050
- src/main/decaf/lang/Throwable.h, 4051
- src/main/decaf/net/BindException.h, 4052
- src/main/decaf/net/BufferedSocket.h, 4053
- src/main/decaf/net/ConnectException.h, 4054
- src/main/decaf/net/HttpRetryException.h, 4055
- src/main/decaf/net/MalformedURLException.h, 4056
- src/main/decaf/net/NoRouteToHostException.h, 4057
- src/main/decaf/net/PortUnreachableException.h, 4058
- src/main/decaf/net/ProtocolException.h, 4059
- src/main/decaf/net/ServerSocket.h, 4060
- src/main/decaf/net/Socket.h, 4061
- src/main/decaf/net/SocketError.h, 4062
- src/main/decaf/net/SocketException.h, 4063

- src/main/decaf/net/SocketFactory.h, 4064
- src/main/decaf/net/SocketInputStream.h, 4065
- src/main/decaf/net/SocketOutputStream.h, 4066
- src/main/decaf/net/SocketTimeoutException.h, 4067
- src/main/decaf/net/TcpSocket.h, 4068
- src/main/decaf/net/UnknownHostException.h, 4069
- src/main/decaf/net/UnknownServiceException.h, 4070
- src/main/decaf/net/URI.h, 4071
- src/main/decaf/net/URISyntaxException.h, 4072
- src/main/decaf/net/URL.h, 4073
- src/main/decaf/net/URLDecoder.h, 4074
- src/main/decaf/net/URLEncoder.h, 4075
- src/main/decaf/nio/Buffer.h, 4076
- src/main/decaf/nio/BufferOverflowException.h, 4077
- src/main/decaf/nio/BufferUnderflowException.h, 4078
- src/main/decaf/nio/ByteBuffer.h, 4079
- src/main/decaf/nio/CharBuffer.h, 4080
- src/main/decaf/nio/DoubleBuffer.h, 4081
- src/main/decaf/nio/FloatBuffer.h, 4082
- src/main/decaf/nio/IntBuffer.h, 4083
- src/main/decaf/nio/InvalidMarkException.h, 4084
- src/main/decaf/nio/LongBuffer.h, 4085
- src/main/decaf/nio/ReadOnlyBufferException.h, 4086
- src/main/decaf/nio/ShortBuffer.h, 4087
- src/main/decaf/security/auth/x500/X500Principal.h, 4088
- src/main/decaf/security/cert/Certificate.h, 4089
- src/main/decaf/security/cert/CertificateEncodingException.h, 4090
- src/main/decaf/security/cert/CertificateException.h, 4091
- src/main/decaf/security/cert/CertificateExpiredException.h, 4092
- src/main/decaf/security/cert/CertificateNotYetValidException.h, 4093
- src/main/decaf/security/cert/CertificateParsingException.h, 4094
- src/main/decaf/security/cert/X509Certificate.h, 4095
- src/main/decaf/security/GeneralSecurityException.h, 4096
- src/main/decaf/security/InvalidKeyException.h, 4097
- src/main/decaf/security/Key.h, 4098
- src/main/decaf/security/KeyException.h, 4099
- src/main/decaf/security/NoSuchAlgorithmException.h, 4100
- src/main/decaf/security/NoSuchProviderException.h, 4101
- src/main/decaf/security/Principal.h, 4102
- src/main/decaf/security/PublicKey.h, 4103
- src/main/decaf/security/SignatureException.h, 4104
- src/main/decaf/security_provider/SecurityProvider.h, 4105
- src/main/decaf/security_provider/SecurityProviderMap.h, 4106
- src/main/decaf/security_provider/SecurityProviderRegistrar.h, 4107
- src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h, 4108
- src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h, 4109
- src/main/decaf/util/AbstractCollection.h, 4110
- src/main/decaf/util/AbstractList.h, 4111
- src/main/decaf/util/AbstractMap.h, 4112
- src/main/decaf/util/AbstractQueue.h, 4113
- src/main/decaf/util/AbstractSequentialList.h, 4114
- src/main/decaf/util/AbstractSet.h, 4115
- src/main/decaf/util/Collection.h, 4116
- src/main/decaf/util/Comparator.h, 4117
- src/main/decaf/util/comparators/Less.h, 4118
- src/main/decaf/util/concurrent/atomic/AtomicBoolean.h, 4119
- src/main/decaf/util/concurrent/atomic/AtomicInteger.h, 4120
- src/main/decaf/util/concurrent/atomic/AtomicReference.h, 4121
- src/main/decaf/util/concurrent/BlockingQueue.h, 4122
- src/main/decaf/util/concurrent/BrokenBarrierException.h, 4123
- src/main/decaf/util/concurrent/Callable.h, 4124
- src/main/decaf/util/concurrent/CancellationException.h, 4125
- src/main/decaf/util/concurrent/Concurrent.h, 4126
- src/main/decaf/util/concurrent/ConcurrentMap.h, 4127
- src/main/decaf/util/concurrent/ConcurrentStlMap.h, 4128

- src/main/decaf/util/concurrent/CountDownLatch.h, 4129
- src/main/decaf/util/concurrent/Delayed.h, 4130
- src/main/decaf/util/concurrent/ExecutionException.h, 4131
- src/main/decaf/util/concurrent/Executor.h, 4132
- src/main/decaf/util/concurrent/ExecutorService.h, 4133
- src/main/decaf/util/concurrent/Future.h, 4134
- src/main/decaf/util/concurrent/Lock.h, 4135
- src/main/decaf/util/concurrent/locks/Condition.h, 4137
- src/main/decaf/util/concurrent/locks/Lock.h, 4136
- src/main/decaf/util/concurrent/locks/LockSupport.h, 4138
- src/main/decaf/util/concurrent/locks/ReadWriteLock.h, 4139
- src/main/decaf/util/concurrent/locks/ReentrantLock.h, 4140
- src/main/decaf/util/concurrent/Mutex.h, 4141
- src/main/decaf/util/concurrent/PooledThread.h, 4142
- src/main/decaf/util/concurrent/PooledThreadListener.h, 4143
- src/main/decaf/util/concurrent/RejectedExecutionException.h, 4144
- src/main/decaf/util/concurrent/RejectedExecutionHandler.h, 4145
- src/main/decaf/util/concurrent/Semaphore.h, 4146
- src/main/decaf/util/concurrent/Synchronizable.h, 4147
- src/main/decaf/util/concurrent/SynchronousQueue.h, 4148
- src/main/decaf/util/concurrent/TaskListener.h, 4149
- src/main/decaf/util/concurrent/ThreadFactory.h, 4150
- src/main/decaf/util/concurrent/ThreadPool.h, 4151
- src/main/decaf/util/concurrent/TimeoutException.h, 4152
- src/main/decaf/util/concurrent/TimeUnit.h, 4153
- src/main/decaf/util/Config.h, 3591
- src/main/decaf/util/Date.h, 4154
- src/main/decaf/util/Iterator.h, 4155
- src/main/decaf/util/List.h, 4156
- src/main/decaf/util/ListIterator.h, 4157
- src/main/decaf/util/logging/ConsoleHandler.h, 4158
- src/main/decaf/util/logging/Filter.h, 4159
- src/main/decaf/util/logging/Formatter.h, 4160
- src/main/decaf/util/logging/Handler.h, 4161
- src/main/decaf/util/logging/Logger.h, 4162
- src/main/decaf/util/logging/LoggerCommon.h, 4163
- src/main/decaf/util/logging/LoggerDefines.h, 4164
- src/main/decaf/util/logging/LoggerHierarchy.h, 4166
- src/main/decaf/util/logging/LogManager.h, 4167
- src/main/decaf/util/logging/LogRecord.h, 4168
- src/main/decaf/util/logging/LogWriter.h, 4169
- src/main/decaf/util/logging/MarkBlockLogger.h, 4170
- src/main/decaf/util/logging/PropertiesChangeListener.h, 4171
- src/main/decaf/util/logging/SimpleFormatter.h, 4172
- src/main/decaf/util/logging/SimpleLogger.h, 4173
- src/main/decaf/util/logging/StreamHandler.h, 4174
- src/main/decaf/util/Map.h, 4175
- src/main/decaf/util/PriorityQueue.h, 4176
- src/main/decaf/util/Properties.h, 4177
- src/main/decaf/util/Queue.h, 3958
- src/main/decaf/util/Random.h, 4178
- src/main/decaf/util/Set.h, 4179
- src/main/decaf/util/StlList.h, 4180
- src/main/decaf/util/StlMap.h, 4181
- src/main/decaf/util/StlQueue.h, 4182
- src/main/decaf/util/StlSet.h, 4183
- src/main/decaf/util/StringTokenizer.h, 4184
- src/main/decaf/util/Timer.h, 4185
- src/main/decaf/util/TimerTask.h, 4186
- src/main/decaf/util/UUID.h, 4187
- stackTrace
- decaf::lang::Exception, 1580
- StandardOutputStream
- decaf::internal::io::StandardOutputStream,
- 2995
- StandardInputStream
- decaf::internal::io::StandardInputStream,
- 3001
- StandardOutputStream
- decaf::internal::io::StandardOutputStream,
- 3009
- start
- activemq::commands::ConsumerControl,
- 1254
- activemq::core::ActiveMQConnection, 241

- activemq::core::ActiveMQConsumer, 270
- activemq::core::ActiveMQSession, 458
- activemq::core::ActiveMQSessionExecutor, 462
- activemq::core::MessageDispatchChannel, 2228
- activemq::transport::correlator::ResponseCorrelator, 2790
- activemq::transport::failover::FailoverTransport, 1622
- activemq::transport::IOTransport, 1828
- activemq::transport::mock::MockTransport, 2380
- activemq::transport::Transport, 3278
- activemq::transport::TransportFilter, 3287
- activemq::wireformat::openwire::OpenWireFormat, 2465
- cms::Startable, 3014
- startupTransport
 - activemq::core::ActiveMQConnectionSupport, 259
- staticCast
 - decaf::lang::Pointer, 2502
- StaticInitializer
 - activemq::core::ActiveMQConstants::StaticInitializer, 3016
- std, 145
- std::binary_function, 720
- std::less< decaf::lang::Pointer< T > >, 1983
- operator(), 1983
- StlList
 - decaf::util::StlList, 3021
- StlMap
 - decaf::util::StlMap, 3033, 3034
- StlQueue
 - decaf::util::StlQueue, 3045
- StlSet
 - decaf::util::StlSet, 3053
- StompFrame
 - activemq::wireformat::stomp::StompFrame, 3062
- StompHelper
 - activemq::wireformat::stomp::StompHelper, 3067
- StompWireFormat
 - activemq::wireformat::stomp::StompWireFormat, 3072
- StompWireFormatFactory
 - activemq::wireformat::stomp::StompWireFormatFactory, 3075
- stop
 - activemq::commands::ConsumerControl, 1254
 - activemq::core::ActiveMQConnection, 242
 - activemq::core::ActiveMQConsumer, 270
 - activemq::core::ActiveMQSession, 458
 - activemq::core::ActiveMQSessionExecutor, 462
 - activemq::core::MessageDispatchChannel, 2228
 - activemq::transport::failover::FailoverTransport, 1623
 - activemq::transport::IOTransport, 1829
 - activemq::transport::mock::MockTransport, 2380
 - activemq::transport::Transport, 3278
 - activemq::transport::TransportFilter, 3288
 - cms::Stoppable, 3076
 - decaf::util::concurrent::PooledThread, 2519
 - decaf::util::concurrent::StompNegotiator, 2663, 2664
 - StreamHandler
 - decaf::util::logging::StreamHandler, 3078
 - STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2559
 - StringTokenizer
 - decaf::util::StringTokenizer, 3094
 - stringValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2552
 - subscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 2731
 - activemq::commands::SubscriptionInfo, 3101
 - SUBSCRIBE
 - activemq::wireformat::stomp::StompCommandConstants, 3059
 - subscribedDestination
 - activemq::commands::SubscriptionInfo, 3101
 - SubscriptionInfo
 - activemq::commands::SubscriptionInfo, 3098
 - SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SubscriptionInfo, 3103
 - activemq::wireformat::openwire::marshal::v2::SubscriptionInfo, 3119
 - activemq::wireformat::openwire::marshal::v3::SubscriptionInfo, 3111
 - activemq::wireformat::openwire::marshal::v4::SubscriptionInfo, 3107
 - activemq::wireformat::openwire::marshal::v5::SubscriptionInfo, 3115
 - subscriptionName
 - activemq::commands::ConsumerInfo, 1307

- activemq::commands::JournalTopicAck, 1862
- subSequence
 - decaf::internal::nio::CharArrayBuffer, 997
 - decaf::lang::CharSequence, 1015
 - decaf::nio::CharBuffer, 1011
- suspend
 - activemq::commands::ConnectionControl, 1139
- swap
 - decaf::lang::AtomicRefCounter, 640
 - decaf::lang::Pointer, 2503
- SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 3134
- synchronized
 - Concurrent.h, 4126
- SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 3140
- syncRequest
 - activemq::core::ActiveMQConnection, 242
 - activemq::core::ActiveMQSession, 458
- System
 - decaf::lang::System, 3145
- take
 - decaf::util::concurrent::SynchronousQueue, 3143
- takeRef
 - decaf::internal::nio::ByteArrayPerspective, 912
- takeSession
 - activemq::cmsutil::SessionPool, 2902
- targetConsumerId
 - activemq::commands::Message, 2161
- Task
 - decaf::util::concurrent::ThreadPool, 3177
- TcpSocket
 - decaf::net::TcpSocket, 3153
- TcpTransport
 - activemq::transport::tcp::TcpTransport, 3160, 3161
- TEMP_POSTFIX
 - activemq::commands::ActiveMQDestination, 284
- TEMP_PREFIX
 - activemq::commands::ActiveMQDestination, 284
- TEMP_QUEUE_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 284
- TEMP_TOPIC_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 284
- TEMPORARY_QUEUE
 - cms::Destination, 1480
- TEMPORARY_TOPIC
 - cms::Destination, 1480
- TEMPQUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- TEMPTOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- TEXT
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- text
 - activemq::commands::ActiveMQTextMessage, 577
- ThreadGroup
 - decaf::lang::ThreadGroup, 3174
- ThreadPool
 - decaf::util::concurrent::ThreadPool, 3177
- Throwable
 - decaf::lang::Throwable, 3182
- Throwing
 - decaf::util::logging, 144
- tightMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 704
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1441
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller, 183
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller, 219
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 291
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller, 328
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 351
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller, 392
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 429
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller, 485
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 509
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 534
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 559

activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	2212
584	
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessageMarshaller	2245
608	
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	2271
667	
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	2302
761	
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	2327
789	
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2361
1146	
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2411
1170	
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2491
1197	
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2601
1227	
activemq::wireformat::openwire::marshal::v1::ConsumerGroupMarshaller	2625
1265	
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2650
1290	
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	2709
1315	
activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller	2742
1344	
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2774
1369	
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2808
1404	
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	2863
1503	
activemq::wireformat::openwire::marshal::v1::DiscardResponseMarshaller	2883
1533	
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	2939
1591	
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3104
1685	
activemq::wireformat::openwire::marshal::v1::IntegrationResponseMarshaller	3222
1786	
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3247
1852	
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	3397
1869	
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	3421
1900	
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	195
1924	
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	231
1951	
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	303
1967	
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	340
2011	

activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	1840	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	1840
363		activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	1865
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1865	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	1865
404		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1888
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1888	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller	1888
441		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1912
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1912	activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	1912
497		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1935
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1935	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	1935
521		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1963
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1963	activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	1963
546		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1999
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	1999	activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller	1999
571		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2196
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2196	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2196
596		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2233
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2233	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2233
620		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2259
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2259	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2259
688		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2286
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2286	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2286
773		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2307
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2307	activemq::wireformat::openwire::marshal::v2::MessageMarshaller	2307
801		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2353
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2353	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2353
1158		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2399
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2399	activemq::wireformat::openwire::marshal::v2::NetworkBridgeMarshaller	2399
1182		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2479
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2479	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2479
1209		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2585
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2585	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2585
1235		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2613
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2613	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2613
1273		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2638
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2638	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2638
1298		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2713
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2713	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2713
1327		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2738
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2738	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	2738
1352		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2758
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2758	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	2758
1377		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2793
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2793	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	2793
1416		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2867
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2867	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	2867
1491		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2891
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2891	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	2891
1517		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2951
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2951	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	2951
1587		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	3120
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	3120	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	3120
1681		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	3226
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	3226	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3226
1782		activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	

activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	1361
3263	
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	1400
3389	
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	1495
3429	
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobWireFormatMessageMarshaller	1521
179	
activemq::wireformat::openwire::marshal::v3::ActiveMQByteWireFormatMessageMarshaller	1595
215	
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationFormatMarshaller	1689
287	
activemq::wireformat::openwire::marshal::v3::ActiveMQMapWireFormatMessageMarshaller	1790
324	
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	1848
347	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectWireFormatMessageMarshaller	1873
388	
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueWireFormatMarshaller	1896
425	
activemq::wireformat::openwire::marshal::v3::ActiveMQStackWireFormatMessageMarshaller	1916
481	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueFormatMarshaller	1943
505	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueFormatMarshaller	1971
530	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	2003
555	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextWireFormatMessageMarshaller	2204
580	
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2241
604	
activemq::wireformat::openwire::marshal::v3::BaseCommandWireFormatMarshaller	2267
660	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2294
757	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2317
785	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2357
1142	
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2403
1166	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2483
1193	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2589
1219	
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	2617
1257	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2654
1282	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	2717
1311	
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	2746
1336	

activemq::wireformat::openwire::marshal::v3::ReplyCommandMarshaller	openwire::marshal::v4::ConnectionError
2762	1174
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	openwire::marshal::v4::ConnectionIdMarsh
2798	1201
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	openwire::marshal::v4::ConnectionInfoM
2871	1223
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	openwire::marshal::v4::ConsumerContro
2899	1261
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	openwire::marshal::v4::ConsumerIdMar
2947	1286
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	openwire::marshal::v4::ConsumerInfoMa
3112	1319
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	openwire::marshal::v4::ControlComman
3238	1340
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	openwire::marshal::v4::DataArrayRespo
3255	1365
activemq::wireformat::openwire::marshal::v3::WireFormatMarshaller	openwire::marshal::v4::DataResponseM
3381	1412
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	openwire::marshal::v4::DestinationInfoM
3417	1499
activemq::wireformat::openwire::marshal::v4::ActiveMQBlockWireFormatMarshaller	openwire::marshal::v4::DiscoveryEventM
187	1525
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	openwire::marshal::v4::ExceptionRespo
223	1599
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	openwire::marshal::v4::FlushCommandL
295	1693
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	openwire::marshal::v4::IntegerResponse
332	1794
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	openwire::marshal::v4::JournalQueueAc
355	1844
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	openwire::marshal::v4::JournalTopicAck
396	1877
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMarshaller	openwire::marshal::v4::JournalTraceMa
433	1892
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireFormatMarshaller	openwire::marshal::v4::JournalTransact
489	1920
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	openwire::marshal::v4::KeepAliveInfoM
513	1947
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	openwire::marshal::v4::LastPartialComr
538	1975
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	openwire::marshal::v4::LocalTransaction
563	2007
activemq::wireformat::openwire::marshal::v4::ActiveMQTextWireFormatMarshaller	openwire::marshal::v4::MessageAckMar
588	2208
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	openwire::marshal::v4::MessageDispatc
612	2237
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	openwire::marshal::v4::MessageDispatc
674	2263
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	openwire::marshal::v4::MessageIdMarsh
765	2290
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	openwire::marshal::v4::MessageMarshal
793	2312
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	openwire::marshal::v4::MessagePullMar
1150	2369

activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	openwire::marshal::v5::ActiveMQTemp5
2415	567
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	openwire::marshal::v5::ActiveMQTextM
2487	592
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::ActiveMQTopicC
2593	616
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::BaseCommandM
2629	681
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	openwire::marshal::v5::BrokerIdMarsh
2646	769
activemq::wireformat::openwire::marshal::v4::RemoteInfoMarshaller	openwire::marshal::v5::BrokerInfoMarsh
2721	797
activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	openwire::marshal::v5::ConnectionCont
2750	1154
activemq::wireformat::openwire::marshal::v4::ReplaceCommandMarshaller	openwire::marshal::v5::ConnectionError
2770	1178
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	openwire::marshal::v5::ConnectionIdMa
2813	1205
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	openwire::marshal::v5::ConnectionInfoM
2875	1231
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	openwire::marshal::v5::ConsumerContro
2887	1269
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	openwire::marshal::v5::ConsumerIdMar
2943	1294
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	openwire::marshal::v5::ConsumerInfoMa
3108	1323
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	openwire::marshal::v5::ControlComm
3230	1348
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	openwire::marshal::v5::DataArrayRespo
3251	1373
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	openwire::marshal::v5::DataResponseM
3393	1408
activemq::wireformat::openwire::marshal::v4::XATransactionMarshaller	openwire::marshal::v5::DestinationInfoM
3425	1507
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	openwire::marshal::v5::DiscoveryEventM
191	1529
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	openwire::marshal::v5::ExceptionRespo
227	1603
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	openwire::marshal::v5::FlushCommandL
299	1697
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	openwire::marshal::v5::IntegerResponse
336	1798
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	openwire::marshal::v5::JournalQueueAc
359	1856
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	openwire::marshal::v5::JournalTopicAck
400	1881
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	openwire::marshal::v5::JournalTraceMa
437	1904
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaller	openwire::marshal::v5::JournalTransact
493	1928
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	openwire::marshal::v5::KeepAliveInfoM
517	1939
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	openwire::marshal::v5::LastPartialCom
542	1979

1533	2883
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller
1591	2939
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller
1685	3104
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller
1786	3222
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller
1852	3247
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller
1869	3397
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller
1900	3421
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller
1924	195
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller
1951	231
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
1967	303
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller
2011	340
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
2212	363
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller
2245	404
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
2271	441
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller
2302	497
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2327	521
activemq::wireformat::openwire::marshal::v1::MessagePublishMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2361	546
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFileMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2411	571
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMarshaller
2491	596
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
2601	620
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
2625	689
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller
2650	773
activemq::wireformat::openwire::marshal::v1::RemoveFromQueueMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller
2709	801
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionContainerMarshaller
2742	1158
activemq::wireformat::openwire::marshal::v1::ReplyCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller
2774	1182
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller
2809	1209
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller
2863	1235
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerControllerMarshaller

1273	2613
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller
1298	2638
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller
1327	2713
activemq::wireformat::openwire::marshal::v2::ConsumerQueueIdMarshaller	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller
1352	2738
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
1377	2758
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller
1416	2794
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller
1491	2867
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller
1517	2891
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller
1587	2951
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller
1681	3120
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller
1782	3226
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller
1840	3263
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller
1865	3389
activemq::wireformat::openwire::marshal::v2::JournalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller
1888	3429
activemq::wireformat::openwire::marshal::v2::JournalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller
1912	179
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller
1935	215
activemq::wireformat::openwire::marshal::v2::LastReceivedCommandMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
1963	287
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller
1999	324
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
2196	347
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller
2233	388
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
2259	425
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller
2286	481
activemq::wireformat::openwire::marshal::v2::MessageMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTemplateMarshaller
2307	505
activemq::wireformat::openwire::marshal::v2::MessagePushMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller
2353	530
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller
2399	555
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMarshaller
2479	580
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
2585	604
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller

661	2267
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller
757	2294
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::MessageMarshaller
785	2317
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
1142	2357
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFailureMarshaller
1166	2403
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1193	2483
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1219	2589
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1257	2617
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1282	2654
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1311	2717
activemq::wireformat::openwire::marshal::v3::ConsumerQueueMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1336	2746
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1361	2762
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1400	2799
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1495	2871
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1521	2899
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1595	2947
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1689	3112
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller
1790	3238
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
1848	3255
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
1873	3381
activemq::wireformat::openwire::marshal::v3::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::XATransactionInfoMarshaller
1896	3417
activemq::wireformat::openwire::marshal::v3::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller
1916	187
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller
1943	223
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
1971	295
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller
2003	332
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
2204	355
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller
2241	396
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller

433	1892
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMessageMarshaller;marshal::v4::JournalTransact	
489	1920
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueDeflationMarshaller;marshal::v4::KeepAliveInfoM	
513	1947
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatMarshaller;marshal::v4::LastPartialComm	
538	1975
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueTopicMarshaller;marshal::v4::LocalTransaction	
563	2007
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller;marshal::v4::MessageAckMar	
588	2208
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller;openwire::marshal::v4::MessageDispatch	
612	2237
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller;openwire::marshal::v4::MessageDispatch	
675	2263
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller;openwire::marshal::v4::MessageIdMarsh	
765	2290
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller;openwire::marshal::v4::MessageMarshal	
793	2312
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller;openwire::marshal::v4::MessagePullMar	
1150	2369
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller;openwire::marshal::v4::NetworkBridgeF	
1174	2415
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller;openwire::marshal::v4::PartialCommand	
1201	2487
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller;openwire::marshal::v4::ProducerAckMar	
1223	2593
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller;openwire::marshal::v4::ProducerIdMars	
1261	2629
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller;openwire::marshal::v4::ProducerInfoMa	
1286	2646
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller;openwire::marshal::v4::RemoveInfoMars	
1319	2721
activemq::wireformat::openwire::marshal::v4::ContactCommandMarshaller;openwire::marshal::v4::RemoveSubscrip	
1340	2750
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller;openwire::marshal::v4::ReplayCommand	
1365	2770
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller;openwire::marshal::v4::ResponseMarsha	
1412	2814
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller;openwire::marshal::v4::SessionIdMarsha	
1499	2875
activemq::wireformat::openwire::marshal::v4::DiscardEventMarshaller;openwire::marshal::v4::SessionInfoMars	
1525	2887
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller;openwire::marshal::v4::ShutdownInfoMa	
1599	2943
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller;openwire::marshal::v4::SubscriptionInfo	
1693	3108
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller;openwire::marshal::v4::TransactionIdMa	
1794	3230
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller;openwire::marshal::v4::TransactionInfo	
1844	3251
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller;openwire::marshal::v4::WireFormatInfo	
1877	3393
activemq::wireformat::openwire::marshal::v4::JournalTransactMarshaller;openwire::marshal::v4::XATransactionI	

3425	1507
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller; openwire::marshal::v5::DiscoveryEventM	1529
191	1529
activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller; openwire::marshal::v5::ExceptionRespon	1603
227	1603
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller; openwire::marshal::v5::FlushCommandL	1697
299	1697
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller; openwire::marshal::v5::IntegerResponse	1798
336	1798
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller; openwire::marshal::v5::JournalQueueAc	1856
359	1856
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller; openwire::marshal::v5::JournalTopicAck	1881
400	1881
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller; openwire::marshal::v5::JournalTraceMa	1904
437	1904
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceMessageMarshaller; openwire::marshal::v5::JournalTransact	1928
493	1928
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller; openwire::marshal::v5::KeepAliveInfoM	1939
517	1939
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller; openwire::marshal::v5::LastPartialComm	1979
542	1979
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller; openwire::marshal::v5::LocalTransaction	2015
567	2015
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller; openwire::marshal::v5::MessageAckMar	2200
592	2200
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller; openwire::marshal::v5::MessageDispatch	2249
616	2249
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller; openwire::marshal::v5::MessageDispatch	2275
682	2275
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller; openwire::marshal::v5::MessageIdMarsh	2298
769	2298
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller; openwire::marshal::v5::MessageMarshal	2322
797	2322
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller; openwire::marshal::v5::MessagePullMar	2365
1154	2365
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller; openwire::marshal::v5::NetworkBridgeF	2407
1178	2407
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller; openwire::marshal::v5::PartialCommand	2495
1205	2495
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller; openwire::marshal::v5::ProducerAckMar	2597
1231	2597
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller; openwire::marshal::v5::ProducerIdMars	2621
1269	2621
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller; openwire::marshal::v5::ProducerInfoMa	2642
1294	2642
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller; openwire::marshal::v5::RemoveInfoMars	2725
1323	2725
activemq::wireformat::openwire::marshal::v5::ContractCommandMarshaller; openwire::marshal::v5::RemoveSubscrip	2734
1348	2734
activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller; openwire::marshal::v5::ReplayCommand	2766
1373	2766
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller; openwire::marshal::v5::ResponseMarsha	2804
1408	2804
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller; openwire::marshal::v5::SessionIdMarsha	

2859	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller,	709
2895	tightUnmarshal
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller,	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,
2955	709
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	activemq::wireformat::openwire::marshal::DataStreamMarshaller,
3116	1453
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaller,
3234	184
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaller,
3259	220
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,
3385	292
activemq::wireformat::openwire::marshal::v5::XATransactionInfoMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaller,
3433	329
tightMarshalBrokerError1	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	352
704	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaller,
tightMarshalBrokerError2	393
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,
705	430
tightMarshalCachedObject1	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller,
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	486
705	activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller,
tightMarshalCachedObject2	510
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller,
706	539
tightMarshalLong1	activemq::wireformat::openwire::marshal::v1::ActiveMQTemplateMarshaller,
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	560
706	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller,
tightMarshalLong2	585
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,
706	609
tightMarshalNestedObject1	activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller,
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	669
707	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller,
activemq::wireformat::openwire::OpenWireFormat,	762
2458	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller,
tightMarshalNestedObject2	790
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller,
707	1117
activemq::wireformat::openwire::OpenWireFormat,	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller,
2458	1171
tightMarshalObjectArray1	activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller,
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	1198
708	activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller,
tightMarshalObjectArray2	1228
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	activemq::wireformat::openwire::marshal::v1::ConsumerControllerMarshaller,
708	1266
tightMarshalString1	activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller,
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	1291
709	1316
tightMarshalString2	activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller,

1345	2743
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaler	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaler
1370	2775
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaler	activemq::wireformat::openwire::marshal::v1::ResponseMarshaler
1405	2809
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaler	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaler
1504	2864
activemq::wireformat::openwire::marshal::v1::DiscoveryInfoMarshaler	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaler
1534	2884
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaler	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaler
1592	2940
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaler	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaler
1686	3105
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaler	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaler
1787	3223
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaler	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaler
1853	3248
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaler	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaler
1870	3398
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaler	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaler
1901	3422
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaler
1925	196
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaler
1952	232
activemq::wireformat::openwire::marshal::v1::LastBatchCommandMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaler
1968	304
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaler
2012	341
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaler
2213	364
activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaler
2246	405
activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaler
2272	442
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaler
2303	498
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQTemp1Marshaler
2328	522
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQTemp2Marshaler
2362	547
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQTemp3Marshaler
2412	572
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMarshaler
2492	597
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaler	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaler
2602	621
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaler	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaler
2626	690
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaler	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaler
2651	774
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaler	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaler
2710	802
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaler	activemq::wireformat::openwire::marshal::v2::ConnectionContextMarshaler

1159	2354
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller; openwire::marshal::v2::NetworkBridgeF	
1183	2400
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller; openwire::marshal::v2::PartialCommand	
1210	2480
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller; openwire::marshal::v2::ProducerAckMar	
1236	2586
activemq::wireformat::openwire::marshal::v2::ConsumerGroupWithMarshaller; openwire::marshal::v2::ProducerIdMars	
1274	2614
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller; openwire::marshal::v2::ProducerInfoMa	
1299	2639
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller; openwire::marshal::v2::RemoveInfoMars	
1328	2714
activemq::wireformat::openwire::marshal::v2::ContainerCommandMarshaller; openwire::marshal::v2::RemoveSubscrip	
1353	2739
activemq::wireformat::openwire::marshal::v2::DataActiveResponseForMarshaller; openwire::marshal::v2::ReplayCommand	
1378	2759
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller; openwire::marshal::v2::ResponseMarsha	
1417	2794
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller; openwire::marshal::v2::SessionIdMarsha	
1492	2868
activemq::wireformat::openwire::marshal::v2::DisconnectInfoMarshaller; openwire::marshal::v2::SessionInfoMars	
1518	2892
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller; openwire::marshal::v2::ShutdownInfoMa	
1588	2952
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller; openwire::marshal::v2::SubscriptionInfo	
1682	3121
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller; openwire::marshal::v2::TransactionIdMa	
1783	3227
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller; openwire::marshal::v2::TransactionInfo	
1841	3264
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller; openwire::marshal::v2::WireFormatInfo	
1866	3390
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller; openwire::marshal::v2::XATransactionI	
1889	3430
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller; openwire::marshal::v3::ActiveMQBlobM	
1913	180
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller; openwire::marshal::v3::ActiveMQBytesI	
1936	216
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller; openwire::marshal::v3::ActiveMQDestin	
1964	288
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller; openwire::marshal::v3::ActiveMQMapM	
2000	325
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller; openwire::marshal::v3::ActiveMQMessa	
2197	348
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller; openwire::marshal::v3::ActiveMQObject	
2234	389
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller; openwire::marshal::v3::ActiveMQQueue	
2260	426
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller; openwire::marshal::v3::ActiveMQStream	
2287	482
activemq::wireformat::openwire::marshal::v2::MessageMarshaller; openwire::marshal::v3::ActiveMQTempL	
2308	506
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller; openwire::marshal::v3::ActiveMQTemp	

531	1972
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
556	2004
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
581	2205
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
605	2242
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
662	2268
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller
758	2295
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::MessageMarshaller
786	2318
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
1143	2358
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFailureMarshaller
1167	2404
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1194	2484
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1220	2590
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1258	2618
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1283	2655
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1312	2718
activemq::wireformat::openwire::marshal::v3::ContactCommandMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1337	2747
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1362	2763
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1401	2799
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1496	2872
activemq::wireformat::openwire::marshal::v3::DiscardResponseMarshaller	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1522	2900
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1596	2948
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1690	3113
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller
1791	3239
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
1849	3256
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
1874	3382
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller
1897	3418
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller
1917	188
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller
1944	224
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller

- 296
- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller; openwire::marshal::v4::IntegerResponseMarshaller; 1694
- 333
- activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller; openwire::marshal::v4::JournalQueueAckMarshaller; 1795
- 356
- activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller; openwire::marshal::v4::JournalTopicAckMarshaller; 1845
- 397
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller; openwire::marshal::v4::JournalTraceMarshaller; 1878
- 434
- activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceMessageMarshaller; openwire::marshal::v4::JournalTransactionMarshaller; 1893
- 490
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller; openwire::marshal::v4::KeepAliveInfoMarshaller; 1921
- 514
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller; openwire::marshal::v4::LastPartialCommandMarshaller; 1948
- 539
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller; openwire::marshal::v4::LocalTransactionMarshaller; 1976
- 564
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller; openwire::marshal::v4::MessageAckMarshaller; 2008
- 589
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller; openwire::marshal::v4::MessageDispatchMarshaller; 2209
- 613
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller; openwire::marshal::v4::MessageDispatchMarshaller; 2238
- 676
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller; openwire::marshal::v4::MessageIdMarshaller; 2264
- 766
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller; openwire::marshal::v4::MessageMarshaller; 2291
- 794
- activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller; openwire::marshal::v4::MessagePullMarshaller; 2313
- 1151
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller; openwire::marshal::v4::NetworkBridgeMarshaller; 2370
- 1175
- activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller; openwire::marshal::v4::PartialCommandMarshaller; 2416
- 1202
- activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller; openwire::marshal::v4::ProducerAckMarshaller; 2488
- 1224
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller; openwire::marshal::v4::ProducerIdMarshaller; 2594
- 1262
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller; openwire::marshal::v4::ProducerInfoMarshaller; 2630
- 1287
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller; openwire::marshal::v4::RemoveInfoMarshaller; 2647
- 1320
- activemq::wireformat::openwire::marshal::v4::ConsumerQueueMarshaller; openwire::marshal::v4::RemoveSubscriptionMarshaller; 2722
- 1341
- activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller; openwire::marshal::v4::ReplayCommandMarshaller; 2751
- 1366
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller; openwire::marshal::v4::ResponseMarshaller; 2771
- 1413
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller; openwire::marshal::v4::SessionIdMarshaller; 2814
- 1500
- activemq::wireformat::openwire::marshal::v4::DiscardRequestMarshaller; openwire::marshal::v4::SessionInfoMarshaller; 2876
- 1526
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller; openwire::marshal::v4::ShutdownInfoMarshaller; 2888
- 1600
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller; openwire::marshal::v4::SubscriptionInfoMarshaller; 2944

3109	1324
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller
3231	1349
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller
3252	1374
activemq::wireformat::openwire::marshal::v4::WireFormatIdMarshaller	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller
3394	1409
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller
3426	1508
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller
192	1530
activemq::wireformat::openwire::marshal::v5::ActiveMQByteWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller
228	1604
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller
300	1698
activemq::wireformat::openwire::marshal::v5::ActiveMQMapWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller
337	1799
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller
360	1857
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller
401	1882
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller
438	1905
activemq::wireformat::openwire::marshal::v5::ActiveMQSequenceWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller
494	1929
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller
518	1940
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
543	1980
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller
568	2016
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
593	2201
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
617	2250
activemq::wireformat::openwire::marshal::v5::BaseCommandWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
683	2276
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller
770	2299
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v5::MessageMarshaller
798	2323
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller
1155	2366
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v5::NetworkBridgeMarshaller
1179	2408
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller
1206	2496
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller
1232	2598
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller
1270	2622
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller
1295	2643
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller

- 2726 TimeoutException
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoFailureException, 3185, 3186
- 2735
- activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller, 3190
- 2767 decaf::util::Timer, 3190
- activemq::wireformat::openwire::marshal::v5::ResponseMarshallerTask, 3201
- 2804 TimerImpl
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshallerTask, 3201
- 2860 TimerTask
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshallerTask, 3200
- 2896 TimerTaskHeap
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3203
- 2956 timestamp
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshallerMessage, 2161
- 3117 decaf::util::UUID, 3360
- activemq::wireformat::openwire::marshal::v5::TimestampIdMarshaller, 3207
- 3235 decaf::util::concurrent::TimeUnit, 3207
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 417
- 3260 activemq::util::ActiveMQProperties, 417
- activemq::wireformat::openwire::marshal::v5::WireFormatGMSFPersister, 1039
- 3386 decaf::util::AbstractCollection, 157
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 1062
- 3434 decaf::util::concurrent::SynchronousQueue, 3143
- tightUnmarshalBrokerError
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 2664
- 710 decaf::util::StlQueue, 3048
- tightUnmarshalByteArray
- activemq::wireformat::openwire::marshal::BaseBinaryStringMarshaller, 3096
- 710 decaf::lang::Integer, 1772
- tightUnmarshalCachedObject
- activemq::wireformat::openwire::marshal::BaseByteArrayMarshaller, 2067
- 710 decaf::io::ByteArrayOutputStream, 903
- tightUnmarshalConstByteArray
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 904
- 711 toDays
- tightUnmarshalLong
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 3210
- 711 decaf::lang::Math, 2140
- tightUnmarshalNestedObject
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 262
- 712 toHexFromBytes
- activemq::wireformat::openwire::OpenWireFormat, 712
- 2458
- tightUnmarshalString
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1545
- 712 decaf::lang::Float, 1655
- timedJoin
- decaf::util::concurrent::TimeUnit, 3209
- decaf::lang::Integer, 1773
- decaf::lang::Long, 2068
- timedWait
- decaf::util::concurrent::TimeUnit, 3209
- toHours
- decaf::util::concurrent::TimeUnit, 3210
- timeout
- activemq::commands::DestinationInfo, 3211
- 1488
- activemq::commands::MessagePull, 2350
- decaf::util::concurrent::TimeUnit, 3211

- toMinutes
 - decaf::util::concurrent::TimeUnit, 3211
- toNanos
 - decaf::util::concurrent::TimeUnit, 3212
- toOctalString
 - decaf::lang::Integer, 1773
 - decaf::lang::Long, 2068
- TOPIC
 - cms::Destination, 1480
- TOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstant, 3059
- TOPIC_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 284
- toRadians
 - decaf::lang::Math, 2140
- toSeconds
 - decaf::util::concurrent::TimeUnit, 3212
- toStream
 - activemq::wireformat::stomp::StompFrame, 3065
- toString
 - activemq::commands::ActiveMQBlobMessage, 175
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQDestination, 282
 - activemq::commands::ActiveMQMapMessage, 320
 - activemq::commands::ActiveMQMessage, 343
 - activemq::commands::ActiveMQObjectMessage, 385
 - activemq::commands::ActiveMQQueue, 422
 - activemq::commands::ActiveMQStreamMessage, 474
 - activemq::commands::ActiveMQTempDestination, 501
 - activemq::commands::ActiveMQTempQueue, 526
 - activemq::commands::ActiveMQTempTopic, 551
 - activemq::commands::ActiveMQTextMessage, 577
 - activemq::commands::ActiveMQTopic, 601
 - activemq::commands::BaseCommand, 655
 - activemq::commands::BaseDataStructure, 718
 - activemq::commands::BooleanExpression, 738
 - activemq::commands::BrokerId, 753
 - activemq::commands::BrokerInfo, 780
 - activemq::commands::Command, 1066
 - activemq::commands::ConnectionControl, 1138
 - activemq::commands::ConnectionError, 1162
 - activemq::commands::ConnectionId, 1189
 - activemq::commands::ConnectionInfo, 1215
 - activemq::commands::ConsumerControl, 1253
 - activemq::commands::ConsumerId, 1278
 - activemq::commands::ConsumerInfo, 1305
 - activemq::commands::ControlCommand, 1332
 - activemq::commands::DataArrayResponse, 1358
 - activemq::commands::DataResponse, 1397
 - activemq::commands::DataStructure, 1465
 - activemq::commands::DestinationInfo, 1487
 - activemq::commands::DiscoveryEvent, 1513
 - activemq::commands::ExceptionResponse, 1584
 - activemq::commands::FlushCommand, 1678
 - activemq::commands::IntegerResponse, 1779
 - activemq::commands::JournalQueueAck, 1836
 - activemq::commands::JournalTopicAck, 1861
 - activemq::commands::JournalTrace, 1885
 - activemq::commands::JournalTransaction, 1908
 - activemq::commands::KeepAliveInfo, 1932
 - activemq::commands::LastPartialCommand, 1960
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::Message, 2159
 - activemq::commands::MessageAck, 2192
 - activemq::commands::MessageDispatch, 2222
 - activemq::commands::MessageDispatchNotification, 2255
 - activemq::commands::MessageId, 2282
 - activemq::commands::MessagePull, 2349
 - activemq::commands::NetworkBridgeFilter, 2395
 - activemq::commands::PartialCommand, 2475
 - activemq::commands::ProducerAck, 2581

- activemq::commands::ProducerId, 2609
- activemq::commands::ProducerInfo, 2634
- activemq::commands::RemoveInfo, 2705
- activemq::commands::RemoveSubscriptionInfo, 2730
- activemq::commands::ReplayCommand, 2754
- activemq::commands::Response, 2783
- activemq::commands::SessionId, 2856
- activemq::commands::SessionInfo, 2879
- activemq::commands::ShutdownInfo, 2936
- activemq::commands::SubscriptionInfo, 3100
- activemq::commands::TransactionId, 3219
- activemq::commands::TransactionInfo, 3243
- activemq::commands::WireFormatInfo, 3378
- activemq::commands::XATransactionId, 3413
- activemq::core::ActiveMQConstants, 262
- activemq::state::ConnectionState, 1243
- activemq::state::ConsumerState, 1329
- activemq::state::ProducerState, 2656
- activemq::state::SessionState, 2904
- activemq::state::TransactionState, 3266
- activemq::util::ActiveMQProperties, 417
- activemq::util::PrimitiveList, 2535
- activemq::util::PrimitiveMap, 2545
- activemq::util::PrimitiveValueNode, 2567
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 713
- cms::CMSProperties, 1039
- decaf::io::ByteArrayOutputStream, 904
- decaf::lang::Boolean, 735, 736
- decaf::lang::Byte, 846
- decaf::lang::Character, 988
- decaf::lang::CharSequence, 1015
- decaf::lang::Double, 1546
- decaf::lang::Float, 1656
- decaf::lang::Integer, 1774
- decaf::lang::Long, 2068, 2069
- decaf::lang::Short, 2912
- decaf::net::URI, 3318
- decaf::nio::ByteBuffer, 936
- decaf::nio::CharBuffer, 1012
- decaf::nio::DoubleBuffer, 1565
- decaf::nio::FloatBuffer, 1674
- decaf::nio::IntBuffer, 1760
- decaf::nio::LongBuffer, 2088
- decaf::nio::ShortBuffer, 2932
- decaf::security::cert::Certificate, 970
- decaf::security_-
 - provider::unix::openssl::OpenSSLX500Principal, 1862
- 2441
- decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2446
- decaf::util::concurrent::atomic::AtomicBoolean, 631
- decaf::util::concurrent::atomic::AtomicInteger, 637
- decaf::util::concurrent::atomic::AtomicReference, 643
- decaf::util::concurrent::locks::ReentrantLock, 2696
- decaf::util::concurrent::Semaphore, 2833
- decaf::util::concurrent::TimeUnit, 3212
- decaf::util::Date, 1470
- decaf::util::Properties, 2665
- decaf::util::UUID, 3361
- toURI
 - activemq::util::CompositeData, 1089
- toURIOption
 - activemq::core::ActiveMQConstants, 262
- toURL
 - decaf::net::URI, 3318
- track
 - activemq::state::ConnectionStateTracker, 1249
- trackBack
 - activemq::state::ConnectionStateTracker, 1249
- Tracked
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_BEGIN
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_-
 - COMMITONEPHASE
 - activemq::core::ActiveMQConstants, 262
 - COMMITTWOPHASE
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_END
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_FORGET
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_PREPARE
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_RECOVER
 - activemq::core::ActiveMQConstants, 262
- TRANSACTION_STATE_ROLLBACK
 - activemq::core::ActiveMQConstants, 262
- TransactionId
 - activemq::commands::TransactionId, 3218
- transactionId
 - activemq::commands::JournalTopicAck, 2441

- activemq::commands::JournalTransaction, 1909
- activemq::commands::Message, 2161
- activemq::commands::MessageAck, 2193
- activemq::commands::TransactionInfo, 3243
- TransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3221
 - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3225
 - activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3237
 - activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3229
 - activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3233
- TransactionInfo
 - activemq::commands::TransactionInfo, 3241
- TransactionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3246
 - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3262
 - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3254
 - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3250
 - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3258
- TransactionState
 - activemq::core::ActiveMQConstants, 261
 - activemq::state::TransactionState, 3266
- transfer
 - decaf::internal::util::concurrent::TransferQueue, 3269
 - decaf::internal::util::concurrent::TransferStack, 3271, 3272
- TransferQueue
 - decaf::internal::util::concurrent::TransferQueue, 3268
- TransferStack
 - decaf::internal::util::concurrent::TransferStack, 3271
- TransportFilter
 - activemq::transport::TransportFilter, 3283
- transportInterrupted
 - activemq::core::ActiveMQConnection, 242
 - activemq::transport::DefaultTransportListener, 1476
 - activemq::transport::failover::FailoverTransportListener, 1628
 - activemq::transport::TransportFilter, 3288
- activemq::transport::TransportListener, 3290
- transportResumed
 - activemq::core::ActiveMQConnection, 242
 - activemq::transport::DefaultTransportListener, 1476
 - activemq::transport::failover::FailoverTransportListener, 1628
 - activemq::transport::TransportFilter, 3288
 - activemq::transport::TransportListener, 3290
 - TransactionIdMarshaller, 3221
 - decaf::util::concurrent::Semaphore, 2833–2835
 - tryLock
 - activemq::MessageDispatchChannel, 2228
 - decaf::internal::io::StandardErrorOutputStream, 2996
 - decaf::internal::io::StandardInputStream, 3005
 - decaf::internal::io::StandardOutputStream, 3010
 - decaf::internal::concurrent::SynchronizableImpl, 3135
 - decaf::io::BlockingByteArrayInputStream, 729
 - decaf::io::ByteArrayInputStream, 897
 - decaf::io::ByteArrayOutputStream, 904
 - decaf::io::FilterInputStream, 1637
 - decaf::io::FilterOutputStream, 1643
 - decaf::net::SocketInputStream, 2982
 - decaf::net::SocketOutputStream, 2986
 - decaf::util::AbstractCollection, 157
 - decaf::util::concurrent::ConcurrentStlMap, 1114
 - decaf::util::concurrent::locks::Lock, 2020, 2021
 - decaf::util::concurrent::locks::ReentrantLock, 2696, 2697
 - decaf::util::concurrent::Mutex, 2387
 - decaf::util::concurrent::Synchronizable, 3126
 - decaf::util::StlMap, 3040
 - decaf::util::StlQueue, 3048
 - trylock
 - decaf::internal::util::concurrent::MutexImpl, 2392
 - type
 - activemq::commands::JournalTransaction, 1909
 - activemq::commands::Message, 2161
 - activemq::commands::TransactionInfo, 3243

- UnknownHostException
 - decaf::net::UnknownHostException, 3294, 3295
- UnknownServiceException
 - decaf::net::UnknownServiceException, 3297, 3298
- unlock
 - activemq::core::MessageDispatchChannel, 2229
 - decaf::internal::io::StandardErrorOutputStream, 2997
 - decaf::internal::io::StandardInputStream, 3005
 - decaf::internal::io::StandardOutputStream, 3010
 - decaf::internal::util::concurrent::MutexImpl, 2392
 - decaf::internal::util::concurrent::SynchronizableImpl, 3135
 - decaf::io::BlockingByteArrayInputStream, 730
 - decaf::io::ByteArrayInputStream, 898
 - decaf::io::ByteArrayOutputStream, 904
 - decaf::io::FilterInputStream, 1637
 - decaf::io::FilterOutputStream, 1644
 - decaf::net::SocketInputStream, 2982
 - decaf::net::SocketOutputStream, 2987
 - decaf::util::AbstractCollection, 157
 - decaf::util::concurrent::ConcurrentStlMap, 1115
 - decaf::util::concurrent::Lock, 2024
 - decaf::util::concurrent::locks::Lock, 2021
 - decaf::util::concurrent::locks::ReentrantLock, 2698
 - decaf::util::concurrent::Mutex, 2387
 - decaf::util::concurrent::Synchronizable, 3127
 - decaf::util::StlMap, 3040
 - decaf::util::StlQueue, 3048
- unmarshal
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2549
 - activemq::wireformat::openwire::OpenWireFormat, 2459
 - activemq::wireformat::openwire::utils::BooleanStream, 740
 - activemq::wireformat::stomp::StompWireFormat, 3073
 - activemq::wireformat::WireFormat, 3365
- unmarshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2549
- unmarshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2550
- unmarshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2550
- unpark
 - decaf::util::concurrent::locks::LockSupport, 2027
- unregisterFactory
 - activemq::transport::TransportRegistry, 3293
 - activemq::wireformat::WireFormatRegistry, 3402
- unregisterSecurityProvider
 - decaf::security__provider::SecurityProviderMap, 2824
- unsetenv
 - decaf::lang::System, 3147
- UNSUBSCRIBE
 - activemq::wireformat::stomp::StompCommandConstants, 3059
- unsubscribe
 - activemq::cmsutil::PooledSession, 2517
 - activemq::core::ActiveMQSession, 458
 - cms::Session, 2851
- UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 3300, 3301
- UnsupportedOperationException
 - cms::UnsupportedOperationException, 3306
 - decaf::lang::exceptions::UnsupportedOperationException, 3303, 3304
- URI
 - decaf::net::URI, 3310, 3311
- URIEncoderDecoder
 - decaf::internal::net::URIEncoderDecoder, 3319
- URIHelper
 - decaf::internal::net::URIHelper, 3323
- URIParam
 - activemq::core::ActiveMQConstants, 262
- uriParams
 - activemq::core::ActiveMQConstants::StaticInitializer, 3016
- uriParamsMap
 - activemq::core::ActiveMQConstants::StaticInitializer, 3016
- URIPool
 - activemq::transport::failover::URIPool, 3329
- URISyntaxException
 - decaf::net::URISyntaxException, 3335–3337

- URIType
 - decaf::internal::net::URIType, 3341
- URL
 - decaf::net::URL, 3348
- userID
 - activemq::commands::Message, 2161
- userName
 - activemq::commands::ConnectionInfo, 1216
- UTFDataFormatException
 - decaf::io::UTFDataFormatException, 3353, 3354
- UUID
 - decaf::util::UUID, 3357
- validate
 - decaf::internal::net::URLEncoderDecoder, 3320
- validateAuthority
 - decaf::internal::net::URIHelper, 3326
- validateFragment
 - decaf::internal::net::URIHelper, 3326
- validatePath
 - decaf::internal::net::URIHelper, 3326
- validateQuery
 - decaf::internal::net::URIHelper, 3327
- validateScheme
 - decaf::internal::net::URIHelper, 3327
- validateSimple
 - decaf::internal::net::URLEncoderDecoder, 3320
- validateSsp
 - decaf::internal::net::URIHelper, 3327
- validateUserInfo
 - decaf::internal::net::URIHelper, 3328
- value
 - activemq::commands::BrokerId, 753
 - activemq::commands::ConnectionId, 1190
 - activemq::commands::ConsumerId, 1279
 - activemq::commands::LocalTransactionId, 1996
 - activemq::commands::ProducerId, 2610
 - activemq::commands::SessionId, 2856
- valueOf
 - decaf::lang::Boolean, 736
 - decaf::lang::Byte, 847
 - decaf::lang::Character, 988
 - decaf::lang::Double, 1546, 1547
 - decaf::lang::Float, 1656, 1657
 - decaf::lang::Integer, 1774, 1775
 - decaf::lang::Long, 2069, 2070
 - decaf::lang::Short, 2913
 - decaf::util::concurrent::TimeUnit, 3213
- values
 - decaf::util::concurrent::ConcurrentStlMap, 1115
 - decaf::util::concurrent::TimeUnit, 3214
 - decaf::util::Map, 2105
 - decaf::util::StlMap, 3040
- variant
 - decaf::util::UUID, 3361
- verify
 - decaf::security::cert::Certificate, 970
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2446, 2447
- version
 - decaf::util::UUID, 3361
- visit
 - activemq::commands::BrokerError, 748
 - activemq::commands::BrokerInfo, 781
 - activemq::commands::Command, 1067
 - activemq::commands::ConnectionControl, 1138
 - activemq::commands::ConnectionError, 1162
 - activemq::commands::ConnectionInfo, 1215
 - activemq::commands::ConsumerControl, 1253
 - activemq::commands::ConsumerInfo, 1306
 - activemq::commands::ControlCommand, 1332
 - activemq::commands::DestinationInfo, 1487
 - activemq::commands::FlushCommand, 1678
 - activemq::commands::KeepAliveInfo, 1932
 - activemq::commands::Message, 2159
 - activemq::commands::MessageAck, 2192
 - activemq::commands::MessageDispatch, 2223
 - activemq::commands::MessageDispatchNotification, 2255
 - activemq::commands::MessagePull, 2349
 - activemq::commands::ProducerAck, 2581
 - activemq::commands::ProducerInfo, 2635
 - activemq::commands::RemoveInfo, 2705
 - activemq::commands::RemoveSubscriptionInfo, 2730
 - activemq::commands::ReplayCommand, 2754
 - activemq::commands::Response, 2783
 - activemq::commands::SessionInfo, 2879
 - activemq::commands::ShutdownInfo, 2936
 - activemq::commands::TransactionInfo, 3243

- activemq::commands::WireFormatInfo, 3378
- wait
 - activemq::core::MessageDispatchChannel, 2229, 2230
 - decaf::internal::io::StandardErrorOutputStream, 2997, 2998
 - decaf::internal::io::StandardInputStream, 3005, 3006
 - decaf::internal::io::StandardOutputStream, 3011, 3012
 - decaf::internal::util::concurrent::ConditionImpl, 1127
 - decaf::internal::util::concurrent::SynchronizableImpl, 3135, 3136
 - decaf::io::BlockingByteArrayInputStream, 730, 731
 - decaf::io::ByteArrayInputStream, 898, 899
 - decaf::io::ByteArrayOutputStream, 904, 905
 - decaf::io::FilterInputStream, 1638, 1639
 - decaf::io::FilterOutputStream, 1644, 1645
 - decaf::net::SocketInputStream, 2982, 2983
 - decaf::net::SocketOutputStream, 2987, 2988
 - decaf::util::AbstractCollection, 157, 158
 - decaf::util::concurrent::ConcurrentStlMap, 1115, 1116
 - decaf::util::concurrent::Mutex, 2388, 2389
 - decaf::util::concurrent::Synchronizable, 3128, 3130, 3131
 - decaf::util::StlMap, 3040, 3041
 - decaf::util::StlQueue, 3049, 3050
- WAIT_INFINITE
 - Concurrent.h, 4126
- waitForSpace
 - activemq::util::MemoryUsage, 2143
 - activemq::util::Usage, 3352
- wakeup
 - activemq::core::ActiveMQSession, 458
 - activemq::core::ActiveMQSessionExecutor, 463
 - activemq::threads::CompositeTaskRunner, 1094
 - activemq::threads::DedicatedTaskRunner, 1474
 - activemq::threads::TaskRunner, 3150
- Warn
 - decaf::util::logging, 144
- warn
 - decaf::util::logging::Logger, 2035
 - decaf::util::logging::SimpleLogger, 2963
- wasPrepared
- activemq::commands::JournalTransaction, 1909
- what
 - cms::CMSException, 1033
 - decaf::lang::Exception, 1580
- windowSize
- activemq::commands::ProducerInfo, 2635
- WireFormatInfo
 - activemq::commands::WireFormatInfo, 3371
- WireFormatInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::WireFormatInfo, 3396
 - activemq::wireformat::openwire::marshal::v2::WireFormatInfo, 3388
 - activemq::wireformat::openwire::marshal::v3::WireFormatInfo, 3380
 - activemq::wireformat::openwire::marshal::v4::WireFormatInfo, 3392
 - activemq::wireformat::openwire::marshal::v5::WireFormatInfo, 3384
- WireFormatNegotiator
 - activemq::wireformat::WireFormatNegotiator, 3399
- wrap
 - decaf::nio::ByteBuffer, 936
 - decaf::nio::CharBuffer, 1012
 - decaf::nio::DoubleBuffer, 1565, 1566
 - decaf::nio::FloatBuffer, 1674, 1675
 - decaf::nio::IntBuffer, 1760, 1761
 - decaf::nio::LongBuffer, 2089
 - decaf::nio::ShortBuffer, 2932, 2933
- write
 - activemq::io::LoggingOutputStream, 2040, 2041
 - decaf::internal::io::StandardErrorOutputStream, 2998, 2999
 - decaf::internal::io::StandardOutputStream, 3012, 3013
 - decaf::internal::util::ByteArrayAdapter, 869
 - decaf::io::BufferedOutputStream, 815, 816
 - decaf::io::ByteArrayOutputStream, 906
 - decaf::io::DataOutputStream, 1390
 - decaf::io::FilterOutputStream, 1645, 1646
 - decaf::io::OutputStream, 2470, 2471
 - decaf::io::Writer, 3404
 - decaf::net::SocketOutputStream, 2988, 2989
- writeBoolean
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQStreamMessage, 474

- activemq::wireformat::openwire::utils::BooleanStream, 741
- cms::BytesMessage, 947
- cms::StreamMessage, 3089
- decaf::io::DataOutputStream, 1391
- writeByte
 - activemq::commands::ActiveMQBytesMessage, 208
 - activemq::commands::ActiveMQStreamMessage, 474
 - cms::BytesMessage, 948
 - cms::StreamMessage, 3089
 - decaf::io::DataOutputStream, 1391
 - decaf::io::Writer, 3405
- writeBytes
 - activemq::commands::ActiveMQBytesMessage, 209
 - activemq::commands::ActiveMQStreamMessage, 475
 - cms::BytesMessage, 948, 949
 - cms::StreamMessage, 3090
 - decaf::io::DataOutputStream, 1391
- writeChar
 - activemq::commands::ActiveMQBytesMessage, 209
 - activemq::commands::ActiveMQStreamMessage, 475
 - cms::BytesMessage, 949
 - cms::StreamMessage, 3090
 - decaf::io::DataOutputStream, 1391
- writeChars
 - decaf::io::DataOutputStream, 1392
- WriteChecker
 - activemq::transport::inactivity::InactivityMonitor, 1736
 - activemq::transport::inactivity::WriteChecker, 3403
- writeDouble
 - activemq::commands::ActiveMQBytesMessage, 210
 - activemq::commands::ActiveMQStreamMessage, 476
 - cms::BytesMessage, 949
 - cms::StreamMessage, 3091
 - decaf::io::DataOutputStream, 1392
- writeFloat
 - activemq::commands::ActiveMQBytesMessage, 210
 - activemq::commands::ActiveMQStreamMessage, 476
 - cms::BytesMessage, 950
 - cms::StreamMessage, 3091
 - decaf::io::DataOutputStream, 1392
- writeInt
 - activemq::commands::ActiveMQBytesMessage, 210
 - activemq::commands::ActiveMQStreamMessage, 476
 - cms::BytesMessage, 950
 - cms::StreamMessage, 3091
 - decaf::io::DataOutputStream, 1393
- activemq::commands::ActiveMQBytesMessage, 210
- activemq::commands::ActiveMQStreamMessage, 476
- cms::BytesMessage, 950
- cms::StreamMessage, 3091
- decaf::io::DataOutputStream, 1393
- writeLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2689
- writeLong
 - activemq::commands::ActiveMQBytesMessage, 211
 - activemq::commands::ActiveMQStreamMessage, 476
 - cms::BytesMessage, 950
 - cms::StreamMessage, 3092
 - decaf::io::DataOutputStream, 1393
- writeShort
 - activemq::commands::ActiveMQBytesMessage, 211
 - activemq::commands::ActiveMQStreamMessage, 477
 - cms::BytesMessage, 951
 - cms::StreamMessage, 3092
 - decaf::io::DataOutputStream, 1393
- writeString
 - activemq::commands::ActiveMQBytesMessage, 211
 - activemq::commands::ActiveMQStreamMessage, 477
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2469
 - cms::BytesMessage, 951
 - cms::StreamMessage, 3092
- writeTo
 - decaf::io::ByteArrayOutputStream, 907
- writeUnsignedShort
 - activemq::commands::ActiveMQBytesMessage, 211
 - activemq::commands::ActiveMQStreamMessage, 477
 - cms::BytesMessage, 951
 - cms::StreamMessage, 3093
 - decaf::io::DataOutputStream, 1393
- writeUTF
 - activemq::commands::ActiveMQBytesMessage, 212
 - cms::BytesMessage, 952
 - decaf::io::DataOutputStream, 1394
- written
 - decaf::io::DataOutputStream, 1394
- XATransactionId

activemq::commands::XATransactionId,
3411

XATransactionIdMarshaller

activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller,
3420

activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller,
3428

activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller,
3416

activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,
3424

activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller,
3432